# Preface

## Fundamental safety instructions

## A2: Various NC/PLC interface signals and functions

## A3: Axis Monitoring, Protection Zones

## B1: Continuous-path mode, Exact stop, Look Ahead

## B2: Acceleration

## F1: Travel to fixed stop

## G2: Velocities, setpoint / actual value systems, closed-loop control

## H2: Auxiliary function outputs to PLC

## K1: Mode group, channel, program operation, reset response

## K2: Axis Types, Coordinate Systems, Frames

## N2: Emergency stop

## P1: Transverse axes

## P3: Basic PLC program for SINUMERIK 840D sl

## P4: PLC for SINUMERIK 828D

## R1: Referencing

## S1: Spindles

## V1: Feedrates

## W1: Tool offset

## Z1: NC/PLC interface signals

# Appendix

## A: CNC software

### Controllers

- SINUMERIK 840D sl / 840DE sl
- SINUMERIK 828D

### Software

- CNC software: Version 4.7 SP1

---

**01/2015**

6FC5397-0BP40-5BA2
This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

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

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

**CAUTION**
indicates that minor personal injury can result if proper precautions are not taken.

**NOTICE**
indicates that property damage can result if proper precautions are not taken.

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

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

**Proper use of Siemens products**
Note the following:

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

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

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

SINUMERIK documentation
The SINUMERIK documentation is organized in the following categories:

- General documentation
- User documentation
- Manufacturer/service documentation

Additional information
You can find information on the following topics at www.siemens.com/motioncontrol/docu:

- Ordering documentation/overview of documentation
- Additional links to download documents
- Using documentation online (find and search in manuals/information)

Please send any questions about the technical documentation (e.g. suggestions for improvement, corrections) to the following address:

docu.motioncontrol@siemens.com

My Documentation Manager (MDM)
Under the following link you will find information to individually compile OEM-specific machine documentation based on the Siemens content:

www.siemens.com/mdm

Training
For information about the range of training courses, refer under:

- www.siemens.com/sitrain
  SITRAIN - Siemens training for products, systems and solutions in automation technology
- www.siemens.com/sinutrain
  SinuTrain - training software for SINUMERIK

FAQs
SINUMERIK

You can find information on SINUMERIK under the following link:
www.siemens.com/sinumerik

Target group

This publication is intended for:
- Project engineers
- Technologists (from machine manufacturers)
- System startup engineers (Systems/Machines)
- Programmers

Benefits

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

Standard version

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

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

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

Technical Support

You will find telephone numbers for other countries for technical support in the Internet under http://www.siemens.com/automation/service&support

Information on the structure and contents

Installation

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

Note
For detailed descriptions of data and alarms see:
• For machine and setting data:
  Detailed description of machine data (only electronically on DOConCD or DOConWEB)
• For NC/PLC interface signals:
  – Function Manual, Basic Functions; NC/PLC Interface Signals (Z1)
  – Function Manual, Basic Functions; NC/PLC Interface Signals (Z2)
  – Function Manual, Special Functions; NC/PLC Interface Signals (Z3)
• For alarms:
  Diagnostics Manual

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

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

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

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

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

Data types

The control provides the following data types that can be used for programming in part programs:

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
<th>Value range</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>Signed integers</td>
<td>-2,147,483,648 ... +2,147,483,647</td>
</tr>
<tr>
<td>REAL</td>
<td>Numbers with decimal point</td>
<td>≈ ±5,0<em>10⁻³² ≈ ±1,7</em>10⁻³⁰</td>
</tr>
<tr>
<td>BOOL</td>
<td>Boolean values</td>
<td>TRUE (≠0), FALSE (0)</td>
</tr>
<tr>
<td>CHAR</td>
<td>ASCII characters and bytes</td>
<td>0 ... 255 or -128 ... 127</td>
</tr>
<tr>
<td>STRING</td>
<td>Character string, null-terminated</td>
<td>Maximum of 400 characters + /0 (no special characters)</td>
</tr>
<tr>
<td>AXIS</td>
<td>Axis names</td>
<td>All axis names available in the control system</td>
</tr>
<tr>
<td>FRAME</td>
<td>Geometrical parameters for moving, rotating, scaling, and mirroring</td>
<td>---</td>
</tr>
</tbody>
</table>

**Note**

Arrays can only be formed from similar elementary data types. Up to 3-dimensional arrays are possible.

**Querying REAL variables**

We recommend that querying REAL or DOUBLE variables in NC programs and synchronized actions is programmed as limit value evaluation.

**Example:** Querying the actual value of an axis for a specific value

```plaintext
DEF REAL AXPOS = 123.456
IF ($VA_IM[<axis>] - 1ex-6) <= AXPOS <= ($VA_IM[<axis>] + 1ex-6); actual position
  ...< AXPOS
ELSE
  ...<> AXPOS
ENDIF
```
## Table of contents

1. **Preface**........................................................................................................................................ 3
2. **Fundamental safety instructions**.............................................................................................. 33
   1.1 General safety instructions........................................................................................................ 33
   1.2 Industrial security.................................................................................................................. 33
3. **A2: Various NC/PLC interface signals and functions**.............................................................. 35
   2.1 Brief description...................................................................................................................... 35
   2.2 NC/PLC interface signals - only 840D sl............................................................................... 35
      2.2.1 General........................................................................................................................... 35
      2.2.2 Ready signal to PLC........................................................................................................ 37
      2.2.3 Status signals to PLC...................................................................................................... 37
      2.2.4 Signals to/from the operator panel front.......................................................................... 38
      2.2.5 Signals to channel............................................................................................................ 40
      2.2.6 Signals to axis/spindle...................................................................................................... 40
      2.2.7 Signals from axis/spindle............................................................................................... 48
      2.2.8 Signals to axis/spindle (digital drives)............................................................................ 50
      2.2.9 Signals from axis/spindle (digital drives)........................................................................ 51
   2.3 Functions.................................................................................................................................... 53
      2.3.1 Screen refresh behavior for overload - only 840D sl...................................................... 53
      2.3.2 Settings for involute interpolation - only 840D sl............................................................ 54
      2.3.3 Activate DEFAULT memory - only 840D sl.................................................................... 57
      2.3.4 Read and write PLC variable - only 840D sl................................................................... 57
      2.3.5 Access protection via password and keyswitch............................................................... 60
      2.3.5.1 Password.................................................................................................................... 62
      2.3.5.2 Keystwitch positions (DB10, DBX56.4 to 7).............................................................. 63
      2.3.5.3 Parameterizable protection levels.................................................................................. 64
      2.3.6 Switching over motor/drive data sets.............................................................................. 65
      2.3.6.1 General Information.................................................................................................... 65
      2.3.6.2 Formatting interfaces................................................................................................... 65
      2.3.6.3 Request interface......................................................................................................... 66
      2.3.6.4 Display interface.......................................................................................................... 67
      2.3.6.5 Example...................................................................................................................... 67
      2.3.6.6 Overview of the interfaces.......................................................................................... 68
      2.3.6.7 Supplementary conditions........................................................................................... 69
   2.4 Examples..................................................................................................................................... 70
      2.4.1 Parameter set changeover............................................................................................... 70
   2.5 Data lists ..................................................................................................................................... 72
      2.5.1 Machine data.................................................................................................................... 72
      2.5.1.1 Display machine data................................................................................................... 72
      2.5.1.2 NC-specific machine data.......................................................................................... 72
      2.5.1.3 Channelspecific machine data.................................................................................... 72
      2.5.1.4 Axis/spindlespecific machine data.............................................................................. 72
      2.5.2 System variables.............................................................................................................. 74

---

Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2

7
### A3: Axis Monitoring, Protection Zones

- 2.5.3 Signals
  - 2.5.3.1 Signals to NC
  - 2.5.3.2 Signals from NC
  - 2.5.3.3 Signals to operator panel front
  - 2.5.3.4 Signals from operator panel front
  - 2.5.3.5 Signals to channel
  - 2.5.3.6 Signals from channel
  - 2.5.3.7 Signals to axis/spindle
  - 2.5.3.8 Signals from axis/spindle

3.1 Brief description
- 3.1.1 Axis monitoring functions
- 3.1.2 Protection zones

3.2 Axis monitoring functions
- 3.2.1 Contour monitoring
- 3.2.2 Contour error
- 3.2.3 Following-error monitoring
- 3.2.4 Positioning, zero speed and clamping monitoring
- 3.2.5 Correlation between positioning, zero-speed and clamping monitoring
- 3.2.6 Positioning monitoring
- 3.2.7 Zero-speed monitoring
- 3.2.8 Parameter set-dependent exact stop and standstill tolerance

3.2.9 Clamping monitoring
- 3.2.10 Speed-setpoint monitoring
- 3.2.11 Actual-velocity monitoring
- 3.2.12 Measuring system monitoring
- 3.2.13 Encoder-limit-frequency monitoring
- 3.2.14 Plausibility check for absolute encoders
- 3.2.15 Customized error reactions
- 3.2.16 Limit-switch monitoring
- 3.2.17 Hardware limit switch
- 3.2.18 Software limit switch
- 3.2.19 Monitoring of the working area limitation
- 3.2.20 General
- 3.2.21 Working area limitation in BKS
- 3.2.22 Working area limitation in WCS/SZS
- 3.2.23 Parking a machine axis
- 3.2.24 Function
- 3.2.25 Supplementary conditions
- 3.2.26 Example: Changing an attachment head for a direct position measuring system
- 3.2.27 Example: Changing an attachment head for two direct position measuring systems
- 3.2.28 Example: Measuring system switchover when encoders are missing in certain parts of the range

3.3 Protection zones
- 3.3.1 General
- 3.3.2 Types of protection zone
- 3.3.3 Definition via part program instruction
- 3.3.4 Definition as per system variable
- 3.3.5 Activating and deactivating protection zones
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.6</td>
<td>Protection zone violation and temporary enabling of individual protection zones</td>
</tr>
<tr>
<td>3.3.7</td>
<td>Restrictions in protection zones</td>
</tr>
<tr>
<td>3.3.8</td>
<td>Checking for protection zone violation, working area limitation and software limit switches (CALCPOSI)</td>
</tr>
<tr>
<td>3.4</td>
<td>Supplementary conditions</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Axis monitoring functions</td>
</tr>
<tr>
<td>3.5</td>
<td>Examples</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Axis monitoring functions</td>
</tr>
<tr>
<td>3.5.1.1</td>
<td>Working area limitation in WCS/SZS</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Protection zones</td>
</tr>
<tr>
<td>3.5.2.1</td>
<td>Definition and activation of protection zones</td>
</tr>
<tr>
<td>3.6</td>
<td>Data lists</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Machine data</td>
</tr>
<tr>
<td>3.6.1.1</td>
<td>NC-specific machine data</td>
</tr>
<tr>
<td>3.6.1.2</td>
<td>Channelspecific machine data</td>
</tr>
<tr>
<td>3.6.1.3</td>
<td>Axis/spindlespecific machine data</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Setting data</td>
</tr>
<tr>
<td>3.6.2.1</td>
<td>Axis/spindlespecific setting data</td>
</tr>
<tr>
<td>3.6.3</td>
<td>Signals</td>
</tr>
<tr>
<td>3.6.3.1</td>
<td>Signals to channel</td>
</tr>
<tr>
<td>3.6.3.2</td>
<td>Signals from channel</td>
</tr>
<tr>
<td>3.6.3.3</td>
<td>Signals to axis/spindle</td>
</tr>
<tr>
<td>3.6.3.4</td>
<td>Signals from axis/spindle</td>
</tr>
<tr>
<td>3.7</td>
<td>General functionality</td>
</tr>
<tr>
<td>3.7.1</td>
<td>Continuous-path mode</td>
</tr>
<tr>
<td>3.7.2</td>
<td>Exact stop mode</td>
</tr>
<tr>
<td>3.7.3</td>
<td>Dynamic adaptations</td>
</tr>
<tr>
<td>3.7.4</td>
<td>LookAhead</td>
</tr>
<tr>
<td>3.7.5</td>
<td>Standard functionality</td>
</tr>
<tr>
<td>3.7.6</td>
<td>Free-form surface mode: Extension function</td>
</tr>
<tr>
<td>3.8</td>
<td>Dynamic response mode for path interpolation</td>
</tr>
<tr>
<td>3.8.1</td>
<td>Dynamic response mode for path interpolation</td>
</tr>
<tr>
<td>3.8.2</td>
<td>Free-form surface mode: Basic functions</td>
</tr>
<tr>
<td>3.9</td>
<td>Compressor functions</td>
</tr>
<tr>
<td>3.9.1</td>
<td>Compression of blocks for linear and circular movement</td>
</tr>
</tbody>
</table>
### Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5.2</td>
<td>Compression of short spline blocks</td>
<td>230</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Supplementary conditions</td>
<td>231</td>
</tr>
<tr>
<td>4.6</td>
<td>Contour/Orientation tolerance</td>
<td>232</td>
</tr>
<tr>
<td>4.7</td>
<td>Tolerance and compression of G0 blocks</td>
<td>235</td>
</tr>
<tr>
<td>4.8</td>
<td>RESET behavior</td>
<td>238</td>
</tr>
<tr>
<td>4.9</td>
<td>Supplementary conditions</td>
<td>238</td>
</tr>
<tr>
<td>4.9.1</td>
<td>Block change and positioning axes</td>
<td>238</td>
</tr>
<tr>
<td>4.9.2</td>
<td>Block change delay</td>
<td>239</td>
</tr>
<tr>
<td>4.10</td>
<td>Data lists</td>
<td>239</td>
</tr>
<tr>
<td>4.10.1</td>
<td>Machine data</td>
<td>239</td>
</tr>
<tr>
<td>4.10.1.1</td>
<td>Channelspecific machine data</td>
<td>239</td>
</tr>
<tr>
<td>4.10.1.3</td>
<td>Axespindlespecific machine data</td>
<td>240</td>
</tr>
<tr>
<td>4.10.2</td>
<td>Setting data</td>
<td>241</td>
</tr>
<tr>
<td>4.10.2.1</td>
<td>Channelspecific setting data</td>
<td>241</td>
</tr>
<tr>
<td>4.10.3</td>
<td>Signals</td>
<td>241</td>
</tr>
<tr>
<td>4.10.3.1</td>
<td>Signals from channel</td>
<td>241</td>
</tr>
<tr>
<td>4.10.3.2</td>
<td>Signals from axis/spindle</td>
<td>241</td>
</tr>
</tbody>
</table>

#### 5 B2: Acceleration

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Brief description</td>
<td>243</td>
</tr>
<tr>
<td>5.1.1</td>
<td>General</td>
<td>243</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Features</td>
<td>243</td>
</tr>
<tr>
<td>5.2</td>
<td>Functions</td>
<td>245</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Acceleration without jerk limitation (BRISK/BRISKA) (channel/axis-specific)</td>
<td>245</td>
</tr>
<tr>
<td>5.2.1.1</td>
<td>General Information</td>
<td>245</td>
</tr>
<tr>
<td>5.2.1.2</td>
<td>Parameterization</td>
<td>246</td>
</tr>
<tr>
<td>5.2.1.3</td>
<td>Programming</td>
<td>247</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Constant travel time (channel-specific)</td>
<td>248</td>
</tr>
<tr>
<td>5.2.2.1</td>
<td>General Information</td>
<td>248</td>
</tr>
<tr>
<td>5.2.2.2</td>
<td>Parameterization</td>
<td>249</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Acceleration matching (ACC) (axis-specific)</td>
<td>250</td>
</tr>
<tr>
<td>5.2.3.1</td>
<td>General Information</td>
<td>250</td>
</tr>
<tr>
<td>5.2.3.2</td>
<td>Programming</td>
<td>250</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Acceleration margin (channel-specific)</td>
<td>251</td>
</tr>
<tr>
<td>5.2.4.1</td>
<td>General Information</td>
<td>251</td>
</tr>
<tr>
<td>5.2.4.2</td>
<td>Parameterization</td>
<td>251</td>
</tr>
<tr>
<td>5.2.5</td>
<td>Path-acceleration limitation (channel-specific)</td>
<td>251</td>
</tr>
<tr>
<td>5.2.5.1</td>
<td>General Information</td>
<td>251</td>
</tr>
<tr>
<td>5.2.5.2</td>
<td>Parameterization</td>
<td>252</td>
</tr>
<tr>
<td>5.2.5.3</td>
<td>Programming</td>
<td>252</td>
</tr>
<tr>
<td>5.2.6</td>
<td>Path acceleration for real-time events (channel-specific)</td>
<td>253</td>
</tr>
<tr>
<td>5.2.6.1</td>
<td>General Information</td>
<td>253</td>
</tr>
<tr>
<td>5.2.6.2</td>
<td>Programming</td>
<td>254</td>
</tr>
<tr>
<td>5.2.7</td>
<td>Acceleration with programmed rapid traverse (G00) (axis-specific)</td>
<td>255</td>
</tr>
<tr>
<td>5.2.7.1</td>
<td>General Information</td>
<td>255</td>
</tr>
<tr>
<td>5.2.7.2</td>
<td>Parameterization</td>
<td>255</td>
</tr>
<tr>
<td>5.2.8</td>
<td>Acceleration with active jerk limitation (SOFT/SOFTA) (axis-specific)</td>
<td>256</td>
</tr>
<tr>
<td>5.2.8.1</td>
<td>General Information</td>
<td>256</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.2.8.2</td>
<td>Parameterization.</td>
<td>256</td>
</tr>
<tr>
<td>5.2.9</td>
<td>Excessive acceleration for non-tangential block transitions (axis-specific)</td>
<td>256</td>
</tr>
<tr>
<td>5.2.9.1</td>
<td>General Information.</td>
<td>256</td>
</tr>
<tr>
<td>5.2.10</td>
<td>Acceleration margin for radial acceleration (channel-specific).</td>
<td>257</td>
</tr>
<tr>
<td>5.2.10.1</td>
<td>General Information.</td>
<td>257</td>
</tr>
<tr>
<td>5.2.10.2</td>
<td>Parameterization.</td>
<td>258</td>
</tr>
<tr>
<td>5.2.11</td>
<td>Jerk limitation with path interpolation (SOFT) (channel-specific).</td>
<td>259</td>
</tr>
<tr>
<td>5.2.11.1</td>
<td>General Information.</td>
<td>259</td>
</tr>
<tr>
<td>5.2.11.2</td>
<td>Parameterization.</td>
<td>261</td>
</tr>
<tr>
<td>5.2.12</td>
<td>Jerk limitation with single-axis interpolation (SOFTA) (axis-specific).</td>
<td>262</td>
</tr>
<tr>
<td>5.2.12.1</td>
<td>Parameterization.</td>
<td>262</td>
</tr>
<tr>
<td>5.2.12.2</td>
<td>Programming.</td>
<td>263</td>
</tr>
<tr>
<td>5.2.13</td>
<td>Path-jerk limitation (channel-specific).</td>
<td>263</td>
</tr>
<tr>
<td>5.2.13.1</td>
<td>General Information.</td>
<td>263</td>
</tr>
<tr>
<td>5.2.13.2</td>
<td>Parameterization.</td>
<td>264</td>
</tr>
<tr>
<td>5.2.13.3</td>
<td>Programming.</td>
<td>264</td>
</tr>
<tr>
<td>5.2.14</td>
<td>Path jerk for real-time events (channel-specific).</td>
<td>265</td>
</tr>
<tr>
<td>5.2.14.1</td>
<td>General Information.</td>
<td>265</td>
</tr>
<tr>
<td>5.2.14.2</td>
<td>Programming.</td>
<td>266</td>
</tr>
<tr>
<td>5.2.15</td>
<td>Jerk with programmed rapid traverse (G00) (axis-specific).</td>
<td>267</td>
</tr>
<tr>
<td>5.2.15.1</td>
<td>General Information.</td>
<td>267</td>
</tr>
<tr>
<td>5.2.15.2</td>
<td>Parameterization.</td>
<td>267</td>
</tr>
<tr>
<td>5.2.16</td>
<td>Excessive jerk for block transitions without constant curvature (axis-specific).</td>
<td>268</td>
</tr>
<tr>
<td>5.2.16.1</td>
<td>General Information.</td>
<td>268</td>
</tr>
<tr>
<td>5.2.16.2</td>
<td>Parameterization.</td>
<td>268</td>
</tr>
<tr>
<td>5.2.17</td>
<td>Velocity-dependent jerk adaptation (axis-specific).</td>
<td>268</td>
</tr>
<tr>
<td>5.2.18</td>
<td>Jerk filter (axis-specific).</td>
<td>270</td>
</tr>
<tr>
<td>5.2.18.1</td>
<td>General Information.</td>
<td>270</td>
</tr>
<tr>
<td>5.2.18.2</td>
<td>Parameterization.</td>
<td>272</td>
</tr>
<tr>
<td>5.2.19</td>
<td>Knee-shaped acceleration characteristic curve.</td>
<td>274</td>
</tr>
<tr>
<td>5.2.19.1</td>
<td>Adaptation to the motor characteristic curve.</td>
<td>274</td>
</tr>
<tr>
<td>5.2.19.2</td>
<td>Effects on path acceleration</td>
<td>276</td>
</tr>
<tr>
<td>5.2.19.3</td>
<td>Substitute characteristic curve.</td>
<td>276</td>
</tr>
<tr>
<td>5.2.19.4</td>
<td>Parameterization.</td>
<td>279</td>
</tr>
<tr>
<td>5.2.19.5</td>
<td>Programming.</td>
<td>280</td>
</tr>
<tr>
<td>5.2.19.6</td>
<td>Boundary conditions.</td>
<td>281</td>
</tr>
<tr>
<td>5.2.20</td>
<td>Acceleration and jerk for JOG motions.</td>
<td>282</td>
</tr>
<tr>
<td>5.2.20.1</td>
<td>Parameterization.</td>
<td>282</td>
</tr>
<tr>
<td>5.2.20.2</td>
<td>Supplementary conditions.</td>
<td>284</td>
</tr>
<tr>
<td>5.3</td>
<td>Examples</td>
<td>285</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Acceleration</td>
<td>285</td>
</tr>
<tr>
<td>5.3.1.1</td>
<td>Path velocity characteristic.</td>
<td>285</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Jerk</td>
<td>286</td>
</tr>
<tr>
<td>5.3.2.1</td>
<td>Path velocity characteristic.</td>
<td>286</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Acceleration and jerk</td>
<td>288</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Knee-shaped acceleration characteristic curve.</td>
<td>289</td>
</tr>
<tr>
<td>5.3.4.1</td>
<td>Activation</td>
<td>289</td>
</tr>
<tr>
<td>5.4</td>
<td>Data lists</td>
<td>290</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Machine data</td>
<td>290</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.4.1.1</td>
<td>NC-specific machine data</td>
<td>290</td>
</tr>
<tr>
<td>5.4.1.2</td>
<td>Channel-specific machine data</td>
<td>290</td>
</tr>
<tr>
<td>5.4.1.3</td>
<td>Axis/spindle-specific machine data</td>
<td>291</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Setting data</td>
<td>292</td>
</tr>
<tr>
<td>5.4.2.1</td>
<td>Channelspecific setting data</td>
<td>292</td>
</tr>
<tr>
<td>5.4.3</td>
<td>System variables</td>
<td>292</td>
</tr>
<tr>
<td>6</td>
<td><strong>F1: Travel to fixed stop</strong></td>
<td>293</td>
</tr>
<tr>
<td>6.1</td>
<td>Brief description</td>
<td>293</td>
</tr>
<tr>
<td>6.2</td>
<td>Detailed description</td>
<td>294</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Programming</td>
<td>294</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Functional sequence</td>
<td>296</td>
</tr>
<tr>
<td>6.2.2.1</td>
<td>Selection</td>
<td>296</td>
</tr>
<tr>
<td>6.2.2.2</td>
<td>Fixed stop is reached</td>
<td>297</td>
</tr>
<tr>
<td>6.2.2.3</td>
<td>Fixed stop is not reached</td>
<td>299</td>
</tr>
<tr>
<td>6.2.2.4</td>
<td>Deselection</td>
<td>300</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Behavior during block search</td>
<td>302</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Behavior for reset and function abort</td>
<td>305</td>
</tr>
<tr>
<td>6.2.5</td>
<td>Behavior with regard to other functions</td>
<td>306</td>
</tr>
<tr>
<td>6.2.6</td>
<td>Setting data</td>
<td>307</td>
</tr>
<tr>
<td>6.2.7</td>
<td>System variables</td>
<td>308</td>
</tr>
<tr>
<td>6.2.8</td>
<td>Alarms</td>
<td>310</td>
</tr>
<tr>
<td>6.2.9</td>
<td>Travel with limited torque/force FOC</td>
<td>311</td>
</tr>
<tr>
<td>6.3</td>
<td>Examples</td>
<td>314</td>
</tr>
<tr>
<td>6.4</td>
<td>Data lists</td>
<td>315</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Machine data</td>
<td>315</td>
</tr>
<tr>
<td>6.4.1.1</td>
<td>Axis/spindle-specific machine data</td>
<td>315</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Setting data</td>
<td>316</td>
</tr>
<tr>
<td>6.4.2.1</td>
<td>Axis/spindle-specific setting data</td>
<td>316</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Signals</td>
<td>316</td>
</tr>
<tr>
<td>6.4.3.1</td>
<td>Signals to axis/spindle</td>
<td>316</td>
</tr>
<tr>
<td>6.4.3.2</td>
<td>Signals from axis/spindle</td>
<td>316</td>
</tr>
<tr>
<td>7</td>
<td><strong>G2: Velocities, setpoint / actual value systems, closed-loop control</strong></td>
<td>317</td>
</tr>
<tr>
<td>7.1</td>
<td>Brief description</td>
<td>317</td>
</tr>
<tr>
<td>7.2</td>
<td>Velocities, traversing ranges, accuracies</td>
<td>317</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Velocities</td>
<td>317</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Traversing ranges</td>
<td>319</td>
</tr>
<tr>
<td>7.2.3</td>
<td>Positioning accuracy of the control system</td>
<td>320</td>
</tr>
<tr>
<td>7.2.4</td>
<td>Input/display resolution, computational resolution</td>
<td>320</td>
</tr>
<tr>
<td>7.2.5</td>
<td>Scaling of physical quantities of machine and setting data</td>
<td>322</td>
</tr>
<tr>
<td>7.3</td>
<td>Metric/inch measuring system</td>
<td>325</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Conversion of basic system by part program</td>
<td>325</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Manual switchover of the basic system</td>
<td>329</td>
</tr>
<tr>
<td>7.3.3</td>
<td>FGROUP and FGREF</td>
<td>333</td>
</tr>
<tr>
<td>7.4</td>
<td>Setpoint/actual-value system</td>
<td>336</td>
</tr>
<tr>
<td>7.4.1</td>
<td>General</td>
<td>336</td>
</tr>
<tr>
<td>7.4.2</td>
<td>Setpoint and encoder assignment</td>
<td>339</td>
</tr>
<tr>
<td>7.4.3</td>
<td>Adapting the motor/load ratios</td>
<td>342</td>
</tr>
</tbody>
</table>
### Table of contents

#### 8 H2: Auxiliary function outputs to PLC

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>Brief description</td>
<td>377</td>
</tr>
<tr>
<td>8.2</td>
<td>Predefined auxiliary functions</td>
<td>384</td>
</tr>
<tr>
<td>8.3</td>
<td>Userdefined auxiliary functions</td>
<td>404</td>
</tr>
<tr>
<td>8.4</td>
<td>Associated auxiliary functions</td>
<td>408</td>
</tr>
<tr>
<td>8.5</td>
<td>Type-specific output behavior</td>
<td>410</td>
</tr>
<tr>
<td>8.6</td>
<td>Priorities of the output behavior for which parameters have been assigned</td>
<td>411</td>
</tr>
<tr>
<td>8.7</td>
<td>Programming an auxiliary function</td>
<td>412</td>
</tr>
</tbody>
</table>

---

### 7.4.4 Speed setpoint output

#### 7.6 Optimization of the control

- 7.6.1 Position controller, position setpoint filter: Balancing filter | 362
- 7.6.2 Position controller, position setpoint filter: Jerk filter | 365
- 7.6.3 Position controller, position setpoint filter: Phase filter | 368
- 7.6.4 Position controller: injection of positional deviation | 370
- 7.6.5 Position control with proportional-plus-integral-action controller | 371

### 7.5 Closed-loop control

#### 7.5.2 Parameter sets of the position controller | 361

### 7.7 Data lists

#### 7.7.11 Displaying machine data | 374

#### 7.7.12 NC-specific machine data | 374

#### 7.7.13 Channelspecific machine data | 374

#### 7.7.14 Axis/spindlespecific machine data | 375

---

### 7.4.6.1 Machine data of the actual value resolution

#### 7.4.6.3 Example: Linear axis with rotary encoder on motor | 352

#### 7.4.6.4 Example: Linear axis with rotary encoder on the machine | 353

#### 7.4.6.5 Example: Rotary axis with rotary encoder on motor | 354

#### 7.4.6.6 Example: Rotary axis with rotary encoder on the machine | 356

#### 7.4.6.7 Example: Intermediate gear with encoder on the tool | 357

---

### 7.5.1 General

### 7.6 Optimization of the control

- 7.6.1 Position controller, position setpoint filter: Balancing filter | 362
- 7.6.2 Position controller, position setpoint filter: Jerk filter | 365
- 7.6.3 Position controller, position setpoint filter: Phase filter | 368
- 7.6.4 Position controller: injection of positional deviation | 370
- 7.6.5 Position control with proportional-plus-integral-action controller | 371

---

### 7.7 Data lists

#### 7.7.1 Machine data | 374

#### 7.7.1.1 Displaying machine data | 374

#### 7.7.1.2 NC-specific machine data | 374

#### 7.7.1.3 Channelspecific machine data | 374

#### 7.7.1.4 Axis/spindlespecific machine data | 375

---

### 7.4.4 Speed setpoint output

#### 7.4.5 Machine data of the actual value system | 347

### 7.4.6 Actual-value resolution | 348

### 7.4.6.1 Machine data of the actual value resolution | 348

### 7.4.6.2 Example: Linear axis with linear scale | 351

### 7.4.6.3 Example: Linear axis with rotary encoder on motor | 352

### 7.4.6.4 Example: Linear axis with rotary encoder on the machine | 353

### 7.4.6.5 Example: Rotary axis with rotary encoder on motor | 354

### 7.4.6.6 Example: Rotary axis with rotary encoder on the machine | 356

### 7.4.6.7 Example: Intermediate gear with encoder on the tool | 357

---

The page contains a table of contents for a manual, listing sections and their respective pages.
8.8 Programmable output duration..................................................................................414
8.9 Auxiliary function output to the PLC........................................................................415
8.10 Auxiliary functions without block change delay....................................................416
8.11 M function with an implicit preprocessing stop.....................................................417
8.12 Response to overstore............................................................................................417
8.13 Behavior during block search..................................................................................418
8.13.1 Auxiliary function output during type 1, 2, and 4 block searches..........................418
8.13.2 Assignment of an auxiliary function to a number of groups..............................420
8.13.3 Time stamp of the active M auxiliary function......................................................421
8.13.4 Determining the output sequence.........................................................................421
8.13.5 Output suppression of spindle-specific auxiliary functions...............................423
8.13.6 Auxiliary function output with a type 5 block search (SERUPRO).......................426
8.13.7 ASUB at the end of the SERUPRO......................................................................431
8.14 Implicitly output auxiliary functions........................................................................437
8.15 Information options................................................................................................438
8.15.1 Group-specific modal M auxiliary function display.............................................439
8.15.2 Querying system variables..................................................................................440
8.16 Supplementary conditions.....................................................................................442
8.16.1 General constraints.............................................................................................442
8.16.2 Output behavior..................................................................................................443
8.17 Examples................................................................................................................444
8.17.1 Extension of predefined auxiliary functions.......................................................444
8.17.2 Defining auxiliary functions.................................................................................446
8.18 Data lists..................................................................................................................450
8.18.1 Machine data.......................................................................................................450
8.18.1.1 NC-specific machine data..............................................................................450
8.18.1.2 Channelspecific machine data.........................................................................450
8.18.2 Signals................................................................................................................451
8.18.2.1 Signals to channel..........................................................................................451
8.18.2.2 Signals from channel.....................................................................................451
8.18.2.3 Signals to axis/spindle...................................................................................453
8.18.2.4 Signals from axis/spindle..............................................................................454

9 K1: Mode group, channel, program operation, reset response........................................455

9.1 Product brief.............................................................................................................455
9.2 Mode group.............................................................................................................457
9.2.1 Mode group stop..................................................................................................460
9.2.2 Mode group reset................................................................................................460
9.3 Mode types and mode type change.........................................................................461
9.3.1 Monitoring functions and interlocks of the individual modes..............................466
9.3.2 Mode change........................................................................................................466
9.4 Channel....................................................................................................................467
9.4.1 Global start disable for channel...........................................................................470
9.5 Program test............................................................................................................471
9.5.1 Program execution without setpoint outputs.......................................................471
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.5.2</td>
<td>Program execution in single-block mode</td>
</tr>
<tr>
<td>9.5.3</td>
<td>Program execution with dry run feedrate</td>
</tr>
<tr>
<td>9.5.4</td>
<td>Skip part-program blocks</td>
</tr>
<tr>
<td>9.6</td>
<td>Workpiece simulation</td>
</tr>
<tr>
<td>9.7</td>
<td>Block search, types 1, 2, and 4:</td>
</tr>
<tr>
<td>9.7.1</td>
<td>Description of the function</td>
</tr>
<tr>
<td>9.7.2</td>
<td>Block search in connection with other NCK functions</td>
</tr>
<tr>
<td>9.7.2.1</td>
<td>ASUB after and during block search</td>
</tr>
<tr>
<td>9.7.2.2</td>
<td>PLC actions after block search</td>
</tr>
<tr>
<td>9.7.2.3</td>
<td>Spindle functions after block search</td>
</tr>
<tr>
<td>9.7.2.4</td>
<td>Reading system variables for a block search</td>
</tr>
<tr>
<td>9.7.3</td>
<td>Automatic start of an ASUB after a block search</td>
</tr>
<tr>
<td>9.7.4</td>
<td>Cascaded block search</td>
</tr>
<tr>
<td>9.7.5</td>
<td>Examples for block search with calculation</td>
</tr>
<tr>
<td>9.7.6</td>
<td>Supplementary conditions</td>
</tr>
<tr>
<td>9.7.6.1</td>
<td>Compressor functions (COMPON, COMPCURV, COMPCAD)</td>
</tr>
<tr>
<td>9.8</td>
<td>Block search Type 5 (SERUPRO)</td>
</tr>
<tr>
<td>9.8.1</td>
<td>Description of the function</td>
</tr>
<tr>
<td>9.8.2</td>
<td>Repositioning to the contour (REPOS)</td>
</tr>
<tr>
<td>9.8.2.1</td>
<td>Repositioning with controlled REPOS</td>
</tr>
<tr>
<td>9.8.3</td>
<td>Accelerate block search</td>
</tr>
<tr>
<td>9.8.4</td>
<td>SERUPRO ASUB</td>
</tr>
<tr>
<td>9.8.5</td>
<td>Selfactuating SERUPRO</td>
</tr>
<tr>
<td>9.8.6</td>
<td>Locking a program section for &quot;Continue machining at the contour&quot;</td>
</tr>
<tr>
<td>9.8.7</td>
<td>Behavior during POWER ON, mode change and RESET</td>
</tr>
<tr>
<td>9.8.8</td>
<td>Supplementary conditions</td>
</tr>
<tr>
<td>9.8.8.1</td>
<td>STOPRE in the target block</td>
</tr>
<tr>
<td>9.8.8.2</td>
<td>SPOS in target block</td>
</tr>
<tr>
<td>9.8.8.3</td>
<td>Travel to fixed stop (FXS)</td>
</tr>
<tr>
<td>9.8.8.4</td>
<td>Travel with limited torque/force (FOC)</td>
</tr>
<tr>
<td>9.8.8.5</td>
<td>Synchronous spindle</td>
</tr>
<tr>
<td>9.8.8.6</td>
<td>Couplings and master-slave</td>
</tr>
<tr>
<td>9.8.8.7</td>
<td>Axis functions</td>
</tr>
<tr>
<td>9.8.8.8</td>
<td>Gear stage change</td>
</tr>
<tr>
<td>9.8.8.9</td>
<td>Superimposed motion</td>
</tr>
<tr>
<td>9.8.8.10</td>
<td>NC/PLC interface signals</td>
</tr>
<tr>
<td>9.8.8.11</td>
<td>Making the initial settings more flexible</td>
</tr>
<tr>
<td>9.8.8.12</td>
<td>Compressor functions (COMPON, COMPCURV, COMPCAD)</td>
</tr>
<tr>
<td>9.8.9</td>
<td>System variable</td>
</tr>
<tr>
<td>9.9</td>
<td>Program operation</td>
</tr>
<tr>
<td>9.9.1</td>
<td>Initial settings</td>
</tr>
<tr>
<td>9.9.2</td>
<td>Selection and start of part program or part program block</td>
</tr>
<tr>
<td>9.9.3</td>
<td>Part program interruption</td>
</tr>
<tr>
<td>9.9.4</td>
<td>RESET command</td>
</tr>
<tr>
<td>9.9.5</td>
<td>Program status</td>
</tr>
<tr>
<td>9.9.6</td>
<td>Channel status</td>
</tr>
<tr>
<td>9.9.7</td>
<td>Responses to operator or program actions</td>
</tr>
<tr>
<td>9.9.8</td>
<td>Part-Program Start</td>
</tr>
<tr>
<td>9.9.9</td>
<td>Example of a timing diagram for a program run</td>
</tr>
<tr>
<td>9.9.10</td>
<td>Program jumps</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>9.9.10.1</td>
<td>Jump back to start of program</td>
</tr>
<tr>
<td>9.9.11</td>
<td>Program section repetitions</td>
</tr>
<tr>
<td>9.9.11.1</td>
<td>Overview</td>
</tr>
<tr>
<td>9.9.11.2</td>
<td>Individual part program block</td>
</tr>
<tr>
<td>9.9.11.3</td>
<td>A part program section after a start label</td>
</tr>
<tr>
<td>9.9.11.4</td>
<td>A part program section between a start label and end label</td>
</tr>
<tr>
<td>9.9.11.5</td>
<td>A part program section between a Start label and the key word: ENDLABEL</td>
</tr>
<tr>
<td>9.9.12</td>
<td>Event-driven program call (PROG_EVENT)</td>
</tr>
<tr>
<td>9.9.12.1</td>
<td>Function</td>
</tr>
<tr>
<td>9.9.12.2</td>
<td>Parameterization</td>
</tr>
<tr>
<td>9.9.12.3</td>
<td>Programming</td>
</tr>
<tr>
<td>9.9.12.4</td>
<td>Boundary conditions</td>
</tr>
<tr>
<td>9.9.12.5</td>
<td>Examples</td>
</tr>
<tr>
<td>9.9.13</td>
<td>Influencing the Stop events through Stop delay area</td>
</tr>
<tr>
<td>9.10</td>
<td>Asynchronous subprograms (ASUBs), interrupt routines</td>
</tr>
<tr>
<td>9.10.1</td>
<td>Function</td>
</tr>
<tr>
<td>9.10.1.1</td>
<td>General functionality</td>
</tr>
<tr>
<td>9.10.1.2</td>
<td>Sequence of an interrupt routine in program operation</td>
</tr>
<tr>
<td>9.10.1.3</td>
<td>Interrupt routine with REPOSA</td>
</tr>
<tr>
<td>9.10.1.4</td>
<td>NC response</td>
</tr>
<tr>
<td>9.10.2</td>
<td>Parameterization</td>
</tr>
<tr>
<td>9.10.3</td>
<td>Programming</td>
</tr>
<tr>
<td>9.10.4</td>
<td>Restrictions</td>
</tr>
<tr>
<td>9.10.5</td>
<td>Examples</td>
</tr>
<tr>
<td>9.11</td>
<td>User-specific ASUB for RET and REPOSA</td>
</tr>
<tr>
<td>9.11.1</td>
<td>Function</td>
</tr>
<tr>
<td>9.11.2</td>
<td>Parameter assignment</td>
</tr>
<tr>
<td>9.11.3</td>
<td>Programming</td>
</tr>
<tr>
<td>9.12</td>
<td>Perform ASUB start for user alarms</td>
</tr>
<tr>
<td>9.12.1</td>
<td>Function</td>
</tr>
<tr>
<td>9.12.2</td>
<td>Activation</td>
</tr>
<tr>
<td>9.12.3</td>
<td>Examples</td>
</tr>
<tr>
<td>9.12.3.1</td>
<td>User ASUB from reset - example 1</td>
</tr>
<tr>
<td>9.12.3.2</td>
<td>User ASUB from reset - example 2</td>
</tr>
<tr>
<td>9.12.3.3</td>
<td>User ASUB with M0</td>
</tr>
<tr>
<td>9.12.3.4</td>
<td>User ASUB with stop</td>
</tr>
<tr>
<td>9.12.3.5</td>
<td>User ASUB from stopped</td>
</tr>
<tr>
<td>9.13</td>
<td>Single block</td>
</tr>
<tr>
<td>9.13.1</td>
<td>Parameterization</td>
</tr>
<tr>
<td>9.13.2</td>
<td>Programming</td>
</tr>
<tr>
<td>9.13.3</td>
<td>Supplementary conditions</td>
</tr>
<tr>
<td>9.13.4</td>
<td>Single-block behavior in mode group with type A/B</td>
</tr>
<tr>
<td>9.14</td>
<td>Program control</td>
</tr>
<tr>
<td>9.14.1</td>
<td>Function selection (via operator panel front or PLC)</td>
</tr>
<tr>
<td>9.14.2</td>
<td>Activation of skip levels</td>
</tr>
<tr>
<td>9.14.3</td>
<td>Adapting the size of the interpolation buffer</td>
</tr>
<tr>
<td>9.14.4</td>
<td>Program display modes via an additional basic block display</td>
</tr>
<tr>
<td>9.14.5</td>
<td>Basic block display for ShopMill/ShopTurn</td>
</tr>
<tr>
<td>9.14.6</td>
<td>Structure for a DIN block</td>
</tr>
<tr>
<td>9.14.7</td>
<td>Execution from external</td>
</tr>
</tbody>
</table>
10 K2: Axis Types, Coordinate Systems, Frames

10.1 Description

10.1.1 Axes
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.2</td>
<td>Coordinate systems</td>
<td>663</td>
</tr>
<tr>
<td>10.1.3</td>
<td>Frames</td>
<td>664</td>
</tr>
<tr>
<td>10.2</td>
<td>Axes</td>
<td>667</td>
</tr>
<tr>
<td>10.2.1</td>
<td>Overview</td>
<td>667</td>
</tr>
<tr>
<td>10.2.2</td>
<td>Machine axes</td>
<td>669</td>
</tr>
<tr>
<td>10.2.3</td>
<td>Channel axes</td>
<td>670</td>
</tr>
<tr>
<td>10.2.4</td>
<td>Geometry axes</td>
<td>670</td>
</tr>
<tr>
<td>10.2.5</td>
<td>Special axes</td>
<td>670</td>
</tr>
<tr>
<td>10.2.6</td>
<td>Path axes</td>
<td>671</td>
</tr>
<tr>
<td>10.2.7</td>
<td>Positioning axes</td>
<td>671</td>
</tr>
<tr>
<td>10.2.8</td>
<td>Main axes</td>
<td>672</td>
</tr>
<tr>
<td>10.2.9</td>
<td>Synchronized axes</td>
<td>673</td>
</tr>
<tr>
<td>10.2.10</td>
<td>Axis configuration</td>
<td>675</td>
</tr>
<tr>
<td>10.2.11</td>
<td>Link axes</td>
<td>677</td>
</tr>
<tr>
<td>10.3</td>
<td>Zeros and reference points</td>
<td>678</td>
</tr>
<tr>
<td>10.3.1</td>
<td>Reference points in working space</td>
<td>678</td>
</tr>
<tr>
<td>10.3.2</td>
<td>Position of coordinate systems and reference points</td>
<td>680</td>
</tr>
<tr>
<td>10.4</td>
<td>Coordinate systems</td>
<td>681</td>
</tr>
<tr>
<td>10.4.1</td>
<td>Overview</td>
<td>681</td>
</tr>
<tr>
<td>10.4.2</td>
<td>Machine coordinate system (MCS)</td>
<td>685</td>
</tr>
<tr>
<td>10.4.2.1</td>
<td>Actual value setting with loss of the referencing status (PRESETON)</td>
<td>686</td>
</tr>
<tr>
<td>10.4.2.2</td>
<td>Actual value setting without loss of the referencing status (PRESETONS)</td>
<td>691</td>
</tr>
<tr>
<td>10.4.3</td>
<td>Basic coordinate system (BCS)</td>
<td>696</td>
</tr>
<tr>
<td>10.4.4</td>
<td>Additive offsets</td>
<td>698</td>
</tr>
<tr>
<td>10.4.5</td>
<td>Basic zero system (BZS)</td>
<td>700</td>
</tr>
<tr>
<td>10.4.6</td>
<td>Settable zero system (SZS)</td>
<td>702</td>
</tr>
<tr>
<td>10.4.7</td>
<td>Workpiece coordinate system (WCS)</td>
<td>703</td>
</tr>
<tr>
<td>10.5</td>
<td>Frames</td>
<td>704</td>
</tr>
<tr>
<td>10.5.1</td>
<td>Frame types</td>
<td>704</td>
</tr>
<tr>
<td>10.5.2</td>
<td>Frame components</td>
<td>705</td>
</tr>
<tr>
<td>10.5.2.1</td>
<td>Translation</td>
<td>705</td>
</tr>
<tr>
<td>10.5.2.2</td>
<td>Fine offset</td>
<td>706</td>
</tr>
<tr>
<td>10.5.2.3</td>
<td>Rotation Overview (geometry axes only)</td>
<td>707</td>
</tr>
<tr>
<td>10.5.2.4</td>
<td>Rotation with a Euler: ZY&quot;X&quot; convention (RPY angles)</td>
<td>708</td>
</tr>
<tr>
<td>10.5.2.5</td>
<td>Rotation with a Euler: ZX&quot;Z&quot; convention</td>
<td>713</td>
</tr>
<tr>
<td>10.5.2.6</td>
<td>Rotation in any plane</td>
<td>714</td>
</tr>
<tr>
<td>10.5.2.7</td>
<td>Scaling</td>
<td>715</td>
</tr>
<tr>
<td>10.5.2.8</td>
<td>Mirroring</td>
<td>716</td>
</tr>
<tr>
<td>10.5.2.9</td>
<td>Chain operator</td>
<td>716</td>
</tr>
<tr>
<td>10.5.2.10</td>
<td>Programmable axis name</td>
<td>716</td>
</tr>
<tr>
<td>10.5.2.11</td>
<td>Coordinate transformation</td>
<td>718</td>
</tr>
<tr>
<td>10.5.3</td>
<td>Data management frames and active frames</td>
<td>718</td>
</tr>
<tr>
<td>10.5.3.1</td>
<td>Overview</td>
<td>718</td>
</tr>
<tr>
<td>10.5.3.2</td>
<td>Activating data management frames</td>
<td>720</td>
</tr>
<tr>
<td>10.5.3.3</td>
<td>NCU-global and channel-specific frames</td>
<td>721</td>
</tr>
<tr>
<td>10.5.4</td>
<td>Frame chain and coordinate systems</td>
<td>722</td>
</tr>
<tr>
<td>10.5.4.1</td>
<td>Overview</td>
<td>722</td>
</tr>
<tr>
<td>10.5.4.2</td>
<td>Relative coordinate systems</td>
<td>723</td>
</tr>
<tr>
<td>10.5.4.3</td>
<td>Selectable SZS</td>
<td>725</td>
</tr>
</tbody>
</table>
Function Manual, 01/2015, 6FC5397-0BP40-5BA2

Basic Functions

10.5.4.4 Manual traversing of geometry axes either in the WCS or in the SZS
10.5.4.5 Suppression of frames
10.5.5 Frames of the frame chain
10.5.5.1 Overview
10.5.5.2 Settable frames ($P_{UIFR}[<n>]$)
10.5.5.3 Grinding frames $P_{GFR}[<n>]$
10.5.5.4 Channel-specific basic frames $P_{<n>}$
10.5.5.5 NCU-global basic frames $P_{NCBFR}[<n>]$
10.5.5.6 Active complete basic frame $P_{ACTBFRAME}$
10.5.5.7 Programmable frame $P_{PFRAME}$
10.5.5.8 Channelspecific system frames
10.5.6 Implicit frame changes
10.5.6.1 Frames and switchover of geometry axes
10.5.6.2 Frame for selection and deselection of transformations
10.5.6.3 Adapting active frames
10.5.6.4 Mapped Frames
10.5.7 Inverse frame
10.5.7.1 Additive frame in frame chain
10.5.8 Setting zeros, workpiece measuring and tool measuring
10.5.8.1 Setting zeros, workpiece measuring and tool measuring
10.5.8.2 Zero offset external via system frames
10.5.8.3 Toolholder
10.5.9 Subprograms with SAVE attribute (SAVE)
10.5.10 Data backup
10.5.11 Positions in the coordinate system
10.5.12 Control system response
10.5.12.1 POWER ON
10.5.12.2 Mode change
10.5.12.3 RESET, end of part program
10.5.12.4 Part program start
10.5.12.5 Block search
10.5.12.6 REPOS
10.6 Workpiece-related actual value system
10.6.1 Overview
10.6.2 Use of the workpiece-related actual value system
10.6.3 Special reactions
10.7 Restrictions
10.8 Examples
10.8.1 Axes
10.8.2 Coordinate systems
10.8.3 Frames
10.9 Data lists
10.9.1 Machine data
10.9.1.1 Displaying machine data
10.9.1.2 NC-specific machine data
10.9.1.3 Channel-specific machine data
10.9.1.4 Axis/spindlespecific machine data
10.9.2 Setting data

Table of contents

19
# Table of contents

## 11 N2: Emergency stop

- 11.1 Brief Description ......................................................................................... 803
- 11.2 Relevant standards .................................................................................. 803
- 11.3 Emergency stop control elements ......................................................... 804
- 11.4 Emergency stop sequence .................................................................. 805
- 11.5 Emergency stop acknowledgement ................................................. 806
- 11.6 Data lists ................................................................................................ 808
- 11.6.1 Machine data ...................................................................................... 808
- 11.6.1.1 Axis/spindlespecific machine data ............................................... 808
- 11.6.2 Signals ................................................................................................ 809
- 11.6.2.1 Signals to NC ................................................................................ 809
- 11.6.2.2 Signals from NC ........................................................................... 809
- 11.6.2.3 Signals to BAG .......................................................................... 809

## 12 P1: Transverse axes

- 12.1 Function .................................................................................................. 811
- 12.2 Parameterization .................................................................................... 814
- 12.3 Programming ......................................................................................... 817
- 12.4 Supplementary conditions .................................................................. 818
- 12.5 Examples ............................................................................................... 819
- 12.6 Data lists ................................................................................................ 820
- 12.6.1 Machine data ...................................................................................... 820
- 12.6.1.1 Channelspecific machine data .................................................... 820
- 12.6.1.2 Axis/spindlespecific machine data ........................................... 820

## 13 P3: Basic PLC program for SINUMERIK 840D sl

- 13.1 Brief description .................................................................................... 821
- 13.2 Key data of the PLC CPU .................................................................... 823
- 13.3 PLC operating system version ............................................................. 824
- 13.4 PLC mode selector ............................................................................... 824
- 13.5 Reserve resources (timers, counters, FC, FB, DB, I/O) ...................... 825
- 13.6 Commissioning hardware configuration of the PLC CPU ............. 825
- 13.7 Starting up the PLC program ................................................................. 825
- 13.7.1 Installation of the basic program ...................................................... 825
- 13.7.2 Application of the basic program ...................................................... 826
- 13.7.3 Version codes ................................................................................... 827
- 13.7.4 Machine program ............................................................................. 827
### Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.7.5</td>
<td>Data backup</td>
<td>828</td>
</tr>
<tr>
<td>13.7.6</td>
<td>PLC series startup, PLC archive</td>
<td>828</td>
</tr>
<tr>
<td>13.7.7</td>
<td>Software upgrade</td>
<td>831</td>
</tr>
<tr>
<td>13.7.8</td>
<td>I/O modules (FM, CP modules)</td>
<td>831</td>
</tr>
<tr>
<td>13.7.9</td>
<td>Troubleshooting</td>
<td>832</td>
</tr>
<tr>
<td>13.8</td>
<td>Coupling of the PLC CPU</td>
<td>833</td>
</tr>
<tr>
<td>13.8.1</td>
<td>General</td>
<td>833</td>
</tr>
<tr>
<td>13.8.2</td>
<td>Properties of the PLC CPU</td>
<td>833</td>
</tr>
<tr>
<td>13.8.3</td>
<td>Interface with integrated PLC</td>
<td>833</td>
</tr>
<tr>
<td>13.8.4</td>
<td>Diagnostic buffer on PLC</td>
<td>835</td>
</tr>
<tr>
<td>13.9</td>
<td>Interface structure</td>
<td>836</td>
</tr>
<tr>
<td>13.9.1</td>
<td>PLC/NCK interface</td>
<td>836</td>
</tr>
<tr>
<td>13.9.2</td>
<td>Interface PLC/HMI</td>
<td>842</td>
</tr>
<tr>
<td>13.9.3</td>
<td>PLC/MCP/HHU interface</td>
<td>847</td>
</tr>
<tr>
<td>13.10</td>
<td>Structure and functions of the basic program</td>
<td>849</td>
</tr>
<tr>
<td>13.10.1</td>
<td>Startup and synchronization of NCK PLC</td>
<td>850</td>
</tr>
<tr>
<td>13.10.2</td>
<td>Cyclic operation (OB1)</td>
<td>851</td>
</tr>
<tr>
<td>13.10.3</td>
<td>Time-interrupt processing (OB35)</td>
<td>853</td>
</tr>
<tr>
<td>13.10.4</td>
<td>Process-interrupt processing (OB 40)</td>
<td>853</td>
</tr>
<tr>
<td>13.10.5</td>
<td>Diagnostic alarm, module failure processing (OB82, OB86)</td>
<td>854</td>
</tr>
<tr>
<td>13.10.6</td>
<td>Response to NCK failure</td>
<td>855</td>
</tr>
<tr>
<td>13.10.7</td>
<td>Functions of the basic program called from the user program</td>
<td>856</td>
</tr>
<tr>
<td>13.10.8</td>
<td>Symbolic programming of user program with interface DB</td>
<td>858</td>
</tr>
<tr>
<td>13.10.9</td>
<td>M decoding acc. to list</td>
<td>860</td>
</tr>
<tr>
<td>13.10.10</td>
<td>PLC machine data</td>
<td>864</td>
</tr>
<tr>
<td>13.10.11</td>
<td>Configuration machine control panel, handheld unit, direct keys</td>
<td>867</td>
</tr>
<tr>
<td>13.10.12</td>
<td>Switchover of machine control panel, handheld unit</td>
<td>876</td>
</tr>
<tr>
<td>13.11</td>
<td>SPL for Safety Integrated</td>
<td>878</td>
</tr>
<tr>
<td>13.12</td>
<td>Assignment overview</td>
<td>878</td>
</tr>
<tr>
<td>13.12.1</td>
<td>Assignment: NCK/PLC interface</td>
<td>878</td>
</tr>
<tr>
<td>13.12.2</td>
<td>Assignment: FB/FC</td>
<td>879</td>
</tr>
<tr>
<td>13.12.3</td>
<td>Assignment: DB</td>
<td>879</td>
</tr>
<tr>
<td>13.12.4</td>
<td>Assignment: Timers</td>
<td>880</td>
</tr>
<tr>
<td>13.13</td>
<td>PLC functions for HMI</td>
<td>881</td>
</tr>
<tr>
<td>13.13.1</td>
<td>Program selection from the PLC</td>
<td>881</td>
</tr>
<tr>
<td>13.13.2</td>
<td>Activating the key lock</td>
<td>884</td>
</tr>
<tr>
<td>13.13.3</td>
<td>Operating area numbers</td>
<td>884</td>
</tr>
<tr>
<td>13.13.4</td>
<td>Screen numbers</td>
<td>884</td>
</tr>
<tr>
<td>13.13.4.1</td>
<td>Screen numbers: JOG, manual machine</td>
<td>885</td>
</tr>
<tr>
<td>13.13.4.2</td>
<td>Screen numbers: Reference point approach</td>
<td>890</td>
</tr>
<tr>
<td>13.13.4.3</td>
<td>Screen numbers: MDA</td>
<td>890</td>
</tr>
<tr>
<td>13.13.4.4</td>
<td>Screen numbers: AUTOMATIC</td>
<td>890</td>
</tr>
<tr>
<td>13.13.4.5</td>
<td>Screen numbers: Parameters operating area</td>
<td>891</td>
</tr>
<tr>
<td>13.13.4.6</td>
<td>Screen numbers: Program operating area</td>
<td>892</td>
</tr>
<tr>
<td>13.13.4.7</td>
<td>Screen numbers: Program manager operating area</td>
<td>893</td>
</tr>
<tr>
<td>13.13.4.8</td>
<td>Screen numbers: Diagnostics operating area</td>
<td>893</td>
</tr>
<tr>
<td>13.13.5</td>
<td>HMI monitor</td>
<td>893</td>
</tr>
<tr>
<td>13.14</td>
<td>PLC functions for drive components on the integrated PROFIBUS</td>
<td>895</td>
</tr>
<tr>
<td>13.14.1</td>
<td>Overview</td>
<td>895</td>
</tr>
</tbody>
</table>
### Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.14.2</td>
<td>Performing a start-up</td>
<td>895</td>
</tr>
<tr>
<td>13.14.3</td>
<td>Example</td>
<td>896</td>
</tr>
<tr>
<td>13.15</td>
<td>Memory requirements of the basic PLC program</td>
<td>897</td>
</tr>
<tr>
<td>13.16</td>
<td>Basic conditions and NC VAR selector</td>
<td>900</td>
</tr>
<tr>
<td>13.16.1</td>
<td>Supplementary conditions</td>
<td>900</td>
</tr>
<tr>
<td>13.16.1.1</td>
<td>Programming and parameterizing tools</td>
<td>900</td>
</tr>
<tr>
<td>13.16.1.2</td>
<td>SIMATIC documentation required</td>
<td>902</td>
</tr>
<tr>
<td>13.16.1.3</td>
<td>Relevant SINUMERIK documents</td>
<td>902</td>
</tr>
<tr>
<td>13.16.2</td>
<td>NC VAR selector</td>
<td>903</td>
</tr>
<tr>
<td>13.16.2.1</td>
<td>Overview</td>
<td>903</td>
</tr>
<tr>
<td>13.16.2.2</td>
<td>Description of functions</td>
<td>905</td>
</tr>
<tr>
<td>13.16.2.3</td>
<td>Startup, installation</td>
<td>914</td>
</tr>
<tr>
<td>13.17</td>
<td>Block descriptions</td>
<td>914</td>
</tr>
<tr>
<td>13.17.1</td>
<td>FB1: RUN_UP - basic program, start section</td>
<td>914</td>
</tr>
<tr>
<td>13.17.2</td>
<td>FB2: GET - read NC variable</td>
<td>922</td>
</tr>
<tr>
<td>13.17.3</td>
<td>FB3: PUT - write NC variables</td>
<td>930</td>
</tr>
<tr>
<td>13.17.4</td>
<td>FB4: Request PI service</td>
<td>937</td>
</tr>
<tr>
<td>13.17.4.1</td>
<td>FB4: PI SERV - request PI service</td>
<td>937</td>
</tr>
<tr>
<td>13.17.4.2</td>
<td>List of available PI services</td>
<td>941</td>
</tr>
<tr>
<td>13.17.4.3</td>
<td>PI service: ASUB</td>
<td>942</td>
</tr>
<tr>
<td>13.17.4.4</td>
<td>PI service: CANCEL</td>
<td>943</td>
</tr>
<tr>
<td>13.17.4.5</td>
<td>PI service: CONFIG</td>
<td>943</td>
</tr>
<tr>
<td>13.17.4.6</td>
<td>PI service: DIGION</td>
<td>944</td>
</tr>
<tr>
<td>13.17.4.7</td>
<td>PI service: DIGIOF</td>
<td>944</td>
</tr>
<tr>
<td>13.17.4.8</td>
<td>PI service: FINDBL</td>
<td>944</td>
</tr>
<tr>
<td>13.17.4.9</td>
<td>PI service: LOGIN</td>
<td>945</td>
</tr>
<tr>
<td>13.17.4.10</td>
<td>PI service: LOGOUT</td>
<td>945</td>
</tr>
<tr>
<td>13.17.4.11</td>
<td>PI service: NCRESC</td>
<td>945</td>
</tr>
<tr>
<td>13.17.4.12</td>
<td>PI service: SELECT</td>
<td>945</td>
</tr>
<tr>
<td>13.17.4.13</td>
<td>PI service: SETUDT</td>
<td>946</td>
</tr>
<tr>
<td>13.17.4.14</td>
<td>PI service: SETUFR</td>
<td>947</td>
</tr>
<tr>
<td>13.17.4.15</td>
<td>PI service: RETRAC</td>
<td>947</td>
</tr>
<tr>
<td>13.17.4.16</td>
<td>PI service: CRCEDN</td>
<td>948</td>
</tr>
<tr>
<td>13.17.4.17</td>
<td>PI service: CREACE</td>
<td>948</td>
</tr>
<tr>
<td>13.17.4.18</td>
<td>PI service: CREATO</td>
<td>949</td>
</tr>
<tr>
<td>13.17.4.19</td>
<td>PI service: DELECE</td>
<td>949</td>
</tr>
<tr>
<td>13.17.4.20</td>
<td>PI service: DELETO</td>
<td>950</td>
</tr>
<tr>
<td>13.17.4.21</td>
<td>PI service: MMCSEM</td>
<td>950</td>
</tr>
<tr>
<td>13.17.4.22</td>
<td>PI service: TMCRTD</td>
<td>951</td>
</tr>
<tr>
<td>13.17.4.23</td>
<td>PI service: TMFDPL</td>
<td>952</td>
</tr>
<tr>
<td>13.17.4.24</td>
<td>PI service: TMFPBP</td>
<td>953</td>
</tr>
<tr>
<td>13.17.4.25</td>
<td>PI service: TMGETT</td>
<td>955</td>
</tr>
<tr>
<td>13.17.4.26</td>
<td>PI service: TMMVTL</td>
<td>955</td>
</tr>
<tr>
<td>13.17.4.27</td>
<td>PI service: TMPOSM</td>
<td>957</td>
</tr>
<tr>
<td>13.17.4.28</td>
<td>PI service: TMPCIT</td>
<td>958</td>
</tr>
<tr>
<td>13.17.4.29</td>
<td>PI service: TMRASS</td>
<td>958</td>
</tr>
<tr>
<td>13.17.4.30</td>
<td>PI service: TRESMO</td>
<td>958</td>
</tr>
<tr>
<td>13.17.4.31</td>
<td>PI service: TSEARC</td>
<td>959</td>
</tr>
<tr>
<td>13.17.4.32</td>
<td>PI service: TMCRMT</td>
<td>962</td>
</tr>
<tr>
<td>13.17.4.33</td>
<td>PI service: TMDLMT</td>
<td>963</td>
</tr>
<tr>
<td>13.17.4.34</td>
<td>PI service: POSMT</td>
<td>963</td>
</tr>
</tbody>
</table>
Table of contents

13.18 Signal/data descriptions..................................................................................1063

13.18.1 Interface signals NCK/PLC, HMI/PLC, MCP/PLC........................................1063

13.18.2 Decoded M signals......................................................................................1064

13.18.3 G Functions..................................................................................................1065

13.18.4 Message signals in DB 2...............................................................................1066

13.19 Programming tips with STEP 7........................................................................1067

13.19.1 Copying data..................................................................................................1067

13.19.2 ANY and POINTER.......................................................................................1067

13.19.2.1 Use of POINTER and ANY in FC.................................................................1068

13.19.2.2 Use of POINTER and ANY in FB.................................................................1069

13.19.2.3 POINTER or ANY variable for transfer to FC or FB.................................1070

13.19.3 Multiinstance DB..........................................................................................1072

13.19.4 Strings..........................................................................................................1074

13.19.5 Determining offset addresses for data block structures...............................1074

13.19.6 FB calls........................................................................................................1075

13.20 Data lists..........................................................................................................1076

13.20.1 Machine data...............................................................................................1076

13.20.1.1 Display machine data..................................................................................1076

13.20.1.2 NC-specific machine data.........................................................................1077

13.20.1.3 Channelspecific machine data...................................................................1077

13.20.2 Signals...........................................................................................................1077
# 14 PLc for Sinumerik 828D

## Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.20.2.1 Signals from operator panel</td>
<td>1077</td>
</tr>
<tr>
<td><strong>14</strong> P4: PLc for SINUMERIK 828D</td>
<td>1079</td>
</tr>
<tr>
<td>14.1 Overview</td>
<td>1079</td>
</tr>
<tr>
<td>14.1.1 PLC firmware</td>
<td>1079</td>
</tr>
<tr>
<td>14.1.2 PLC user interface</td>
<td>1079</td>
</tr>
<tr>
<td>14.1.2.1 Data that are cyclically exchanged</td>
<td>1081</td>
</tr>
<tr>
<td>14.1.2.2 Alarms and messages</td>
<td>1081</td>
</tr>
<tr>
<td>14.1.2.3 Retentive data</td>
<td>1082</td>
</tr>
<tr>
<td>14.1.2.4 Non-retentive data</td>
<td>1082</td>
</tr>
<tr>
<td>14.1.2.5 PLC machine data</td>
<td>1082</td>
</tr>
<tr>
<td>14.1.3 PLC key data</td>
<td>1082</td>
</tr>
<tr>
<td>14.1.4 PLC I/O, fast onboard inputs/outputs</td>
<td>1083</td>
</tr>
<tr>
<td>14.1.5 PLC Toolbox</td>
<td>1083</td>
</tr>
<tr>
<td>14.1.5.1 Star/delta changeover</td>
<td>1083</td>
</tr>
<tr>
<td>14.2 Fast on-board inputs and outputs</td>
<td>1083</td>
</tr>
<tr>
<td>14.3 Ladder viewer and ladder add-on</td>
<td>1085</td>
</tr>
<tr>
<td>14.3.1 Overview</td>
<td>1085</td>
</tr>
<tr>
<td>14.3.2 Parameterization</td>
<td>1085</td>
</tr>
<tr>
<td>14.4 PLC Programming Tool</td>
<td>1086</td>
</tr>
<tr>
<td>14.5 Data interface</td>
<td>1086</td>
</tr>
<tr>
<td>14.5.1 PLC-NCK interface</td>
<td>1087</td>
</tr>
<tr>
<td>14.5.1.1 Mode signals</td>
<td>1087</td>
</tr>
<tr>
<td>14.5.1.2 NC channel signals</td>
<td>1088</td>
</tr>
<tr>
<td>14.5.1.3 Axis and spindle signals</td>
<td>1089</td>
</tr>
<tr>
<td>14.5.1.4 General NCK signals</td>
<td>1090</td>
</tr>
<tr>
<td>14.5.1.5 Fast data exchange PLC-NCK</td>
<td>1090</td>
</tr>
<tr>
<td>14.5.2 PLC-HMI interface</td>
<td>1091</td>
</tr>
<tr>
<td>14.5.2.1 Operating area numbers</td>
<td>1091</td>
</tr>
<tr>
<td>14.5.2.2 Screen numbers</td>
<td>1092</td>
</tr>
<tr>
<td>14.6 Function interface</td>
<td>1100</td>
</tr>
<tr>
<td>14.6.1 Read/write NC variables</td>
<td>1100</td>
</tr>
<tr>
<td>14.6.1.1 User interface</td>
<td>1100</td>
</tr>
<tr>
<td>14.6.1.2 Job specification</td>
<td>1101</td>
</tr>
<tr>
<td>14.6.1.3 Job management: Start job</td>
<td>1102</td>
</tr>
<tr>
<td>14.6.1.4 Job management: Waiting for end of job</td>
<td>1103</td>
</tr>
<tr>
<td>14.6.1.5 Job management: Job completion</td>
<td>1103</td>
</tr>
<tr>
<td>14.6.1.6 Job management: Flow diagram</td>
<td>1104</td>
</tr>
<tr>
<td>14.6.1.7 Job evaluation</td>
<td>1105</td>
</tr>
<tr>
<td>14.6.1.8 Operable variables</td>
<td>1107</td>
</tr>
<tr>
<td>14.6.1.9 Specifying selected NC variables</td>
<td>1113</td>
</tr>
<tr>
<td>14.6.2 Program instance services (PI services)</td>
<td>1115</td>
</tr>
<tr>
<td>14.6.2.1 User interface</td>
<td>1115</td>
</tr>
<tr>
<td>14.6.2.2 PI services</td>
<td>1117</td>
</tr>
<tr>
<td>14.6.3 PLC user alarms</td>
<td>1120</td>
</tr>
<tr>
<td>14.6.3.1 User interface</td>
<td>1120</td>
</tr>
<tr>
<td>14.6.4 PLC axis control</td>
<td>1124</td>
</tr>
<tr>
<td>14.6.4.1 Overview</td>
<td>1124</td>
</tr>
<tr>
<td>14.6.4.2 User interface: Preparing the NC axis as PLC axis</td>
<td>1125</td>
</tr>
<tr>
<td>14.6.4.3 User interface: Functionality</td>
<td>1126</td>
</tr>
<tr>
<td>14.6.4.4</td>
<td>Spindle positioning</td>
</tr>
<tr>
<td>14.6.4.5</td>
<td>Rotate spindle</td>
</tr>
<tr>
<td>14.6.4.6</td>
<td>Oscillate spindle</td>
</tr>
<tr>
<td>14.6.4.7</td>
<td>Indexing axis</td>
</tr>
<tr>
<td>14.6.4.8</td>
<td>Positioning axis metric</td>
</tr>
<tr>
<td>14.6.4.9</td>
<td>Positioning axis inch</td>
</tr>
<tr>
<td>14.6.4.10</td>
<td>Positioning axis metric with handwheel override</td>
</tr>
<tr>
<td>14.6.4.11</td>
<td>Positioning axis inch with handwheel override</td>
</tr>
<tr>
<td>14.6.4.12</td>
<td>Rotate spindle with automatic gear stage selection</td>
</tr>
<tr>
<td>14.6.4.13</td>
<td>Rotate spindle with constant cutting rate ([\text{m/min}])</td>
</tr>
<tr>
<td>14.6.4.14</td>
<td>Rotate spindle with constant cutting rate ([\text{feet/min}])</td>
</tr>
<tr>
<td>14.6.4.15</td>
<td>Error messages</td>
</tr>
<tr>
<td>14.6.5</td>
<td>Starting ASUBs</td>
</tr>
<tr>
<td>14.6.5.1</td>
<td>General</td>
</tr>
<tr>
<td>14.6.5.2</td>
<td>Job start</td>
</tr>
<tr>
<td>14.6.5.3</td>
<td>Job result</td>
</tr>
<tr>
<td>14.6.5.4</td>
<td>Signal flow</td>
</tr>
</tbody>
</table>

## 15 R1: Referencing

| 15.1 | Brief Description | 1149 |
| 15.2 | Axis specific referencing | 1150 |
| 15.3 | Channel specific referencing | 1152 |
| 15.4 | Reference point approach from part program (G74) | 1153 |
| 15.5 | Referencing with incremental measurement systems | 1154 |
| 15.5.1 | Hardware signals | 1154 |
| 15.5.2 | Zero mark selection with BERO | 1156 |
| 15.5.3 | Time sequence | 1157 |
| 15.5.4 | Phase 1: Traversing to the reference cam | 1158 |
| 15.5.5 | Phase 2: Synchronization with the zero mark | 1160 |
| 15.5.6 | Phase 3: Traversing to the reference point | 1165 |
| 15.6 | Referencing with distance-coded reference marks | 1167 |
| 15.6.1 | General overview | 1167 |
| 15.6.2 | Basic parameter assignment | 1168 |
| 15.6.3 | Time sequence | 1170 |
| 15.6.4 | Phase 1: Travel across the reference marks with synchronization | 1170 |
| 15.6.5 | Phase 2: Traversing to the target point | 1172 |
| 15.7 | Referencing by means of actual value adjustment | 1174 |
| 15.7.1 | Actual value adjustment to the referencing measurement system | 1174 |
| 15.7.2 | Actual value adjustment for measuring systems with distance-coded reference marks | 1175 |
| 15.8 | Referencing in follow-up mode | 1176 |
| 15.9 | Referencing with absolute encoders | 1179 |
| 15.9.1 | Information about the adjustment | 1179 |
| 15.9.2 | Calibration by entering a reference point offset | 1180 |
| 15.9.3 | Adjustment by entering a reference point value | 1181 |
| 15.9.4 | Automatic calibration with probe | 1183 |
| 15.9.5 | Adjustment with BERO | 1185 |
| 15.9.6 | Reference point approach with absolute encoders | 1186 |
| 15.9.7 | Reference point approach in rotary absolute encoders with external zero mark | 1186 |
16.1 Brief Description............................................................................................................. 1199
16.2 Modes............................................................................................................................ 1200
16.2.1 Overview.................................................................................................................. 1200
16.2.2 Mode change............................................................................................................ 1201
16.2.3 Control mode............................................................................................................ 1202
16.2.4 Oscillation mode...................................................................................................... 1205
16.2.5 Positioning mode..................................................................................................... 1205
16.2.5.1 General functionality.......................................................................................... 1205
16.2.5.2 Positioning from rotation................................................................................... 1212
16.2.5.3 Positioning from standstill................................................................................ 1217
16.2.5.4 "Spindle in position" signal for tool change......................................................... 1220
16.2.6 Axis mode................................................................................................................ 1221
16.2.6.1 General functionality........................................................................................ 1221
16.2.6.2 Implicit transition to axis mode......................................................................... 1224
16.2.7 Initial spindle state.................................................................................................. 1227
16.3 Reference / synchronize.............................................................................................. 1228
16.4 Configurable gear adaptation....................................................................................... 1232
16.4.1 Gear stages for spindles and gear change change................................................... 1232
16.4.2 Spindle gear stage 0................................................................................................. 1243
16.4.3 Determining the spindle gear stage........................................................................ 1245
16.4.4 Parameter set selection during gear step change................................................... 1246
16.4.5 Intermediate gear..................................................................................................... 1249
16.4.6 Nonacknowledged gear step change..................................................................... 1250
16.4.7 Gear step change with oscillation mode.................................................................. 1251
16.4.8 Gear stage change at fixed position....................................................................... 1257
16.4.9 Configurable gear step in M70............................................................................... 1263
16.4.10 Suppression of the gear stage change for DryRun, program test and SERUPRO... 1265
16.5 Additional adaptations to the spindle functionality that can be configured.............. 1267
16.6 Selectable spindles.................................................................................................................. 1269
16.7 Programming........................................................................................................................ 1272
16.7.1 Programming from the part program.............................................................................. 1272
16.7.2 Programming via synchronized actions.......................................................................... 1276
16.7.3 Programming spindle controls via PLC with FC18 - only 840D sl.............................. 1276
16.7.4 Special spindle motion via the PLC interface................................................................. 1277
16.7.5 External programming (PLC, HMI).................................................................................. 1282
16.8 Spindle monitoring................................................................................................................ 1283
16.8.1 Permissible speed ranges............................................................................................... 1283
16.8.2 Axis/spindle stationary.................................................................................................... 1284
16.8.3 Spindle in setpoint range............................................................................................... 1284
16.8.4 Minimum / maximum speed of the gear stage............................................................... 1285
16.8.5 Diagnosis of spindle speed limitation............................................................................. 1286
16.8.6 Maximum spindle speed............................................................................................... 1288
16.8.7 Maximum encoder limit frequency................................................................................ 1290
16.8.8 End point monitoring..................................................................................................... 1291
16.8.9 M40: Automatic gear stage selection for speeds outside the configured switching thresholds ...................................................................................................................... 1293
16.9 Spindle with SMI 24 (Weiss spindle)................................................................................... 1294
16.9.1 General Information....................................................................................................... 1294
16.9.2 Sensor data................................................................................................................... 1295
16.9.3 Clamped state............................................................................................................... 1297
16.9.4 Additional drive parameters........................................................................................... 1298
16.10 Supplementary conditions................................................................................................ 1299
16.10.1 Changing control parameters...................................................................................... 1299
16.11 Examples........................................................................................................................... 1299
16.11.1 Automatic gear step selection (M40)............................................................................ 1299
16.12 Data lists............................................................................................................................. 1300
16.12.1 Machine data............................................................................................................... 1300
16.12.1.1 NC-specific machine data....................................................................................... 1300
16.12.1.2 Channelspecific machine data.................................................................................. 1300
16.12.1.3 Axis/spindlespecific machine data........................................................................... 1301
16.12.2 Setting data.................................................................................................................. 1302
16.12.2.1 Channelspecific setting data.................................................................................... 1302
16.12.2.2 Axis/spindle-specific setting data............................................................................ 1303
16.12.3 signals.......................................................................................................................... 1303
16.12.3.1 Signals to axis/spindle............................................................................................ 1303
16.12.3.2 Signals from axis/spindle....................................................................................... 1304
17 V1: Feedrates......................................................................................................................... 1307
17.1 Brief description.................................................................................................................. 1307
17.2 Path feedrate F .................................................................................................................... 1308
17.2.1 Feedrate type G93, G94, G95........................................................................................ 1308
17.2.2 Type of feedrate G96, G961, G962, G97, G971 .............................................................. 1310
17.2.3 Feedrate for thread cutting (G33, G34, G35, G335, G336) ............................................ 1312
17.2.3.1 Feedrate with G33................................................................................................... 1317
17.2.3.2 Programmable run-in and run-out path for G33, G34 and G35 ............................... 1318
17.2.3.3 Linear increasing/decreasing thread pitch change with G34 and G35 ......................... 1320
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.2.3.4</td>
<td>Fast retraction during thread cutting</td>
<td>1322</td>
</tr>
<tr>
<td>17.2.3.5</td>
<td>Convex thread (G335, G336)</td>
<td>1326</td>
</tr>
<tr>
<td>17.2.4</td>
<td>Feedrate for tapping without compensating chuck (G331, G332)</td>
<td>1331</td>
</tr>
<tr>
<td>17.2.5</td>
<td>Feedrate for tapping with compensating chuck (G63)</td>
<td>1332</td>
</tr>
<tr>
<td>17.3</td>
<td>Feedrate for positioning axes (FA)</td>
<td>1333</td>
</tr>
<tr>
<td>17.4</td>
<td>Feedrate control</td>
<td>1334</td>
</tr>
<tr>
<td>17.4.1</td>
<td>Feedrate disable and feedrate/spindle stop</td>
<td>1334</td>
</tr>
<tr>
<td>17.4.2</td>
<td>Feedrate override on machine control panel</td>
<td>1335</td>
</tr>
<tr>
<td>17.4.3</td>
<td>Programmable feedrate override</td>
<td>1339</td>
</tr>
<tr>
<td>17.4.4</td>
<td>Dry run feedrate</td>
<td>1340</td>
</tr>
<tr>
<td>17.4.5</td>
<td>Multiple feedrate values in one block</td>
<td>1342</td>
</tr>
<tr>
<td>17.4.6</td>
<td>Fixed feedrate values</td>
<td>1347</td>
</tr>
<tr>
<td>17.4.7</td>
<td>Programmable feedrate characteristics</td>
<td>1349</td>
</tr>
<tr>
<td>17.4.8</td>
<td>Feedrate for chamfer/rounding FRC, FRCM</td>
<td>1350</td>
</tr>
<tr>
<td>17.4.9</td>
<td>Non-modal feedrate FB</td>
<td>1352</td>
</tr>
<tr>
<td>17.4.10</td>
<td>Influencing the single axis dynamic response</td>
<td>1353</td>
</tr>
<tr>
<td>17.5</td>
<td>Supplementary conditions</td>
<td>1358</td>
</tr>
<tr>
<td>17.6</td>
<td>Data lists</td>
<td>1359</td>
</tr>
<tr>
<td>17.6.1</td>
<td>Machine data</td>
<td>1359</td>
</tr>
<tr>
<td>17.6.1.1</td>
<td>NC-specific machine data</td>
<td>1359</td>
</tr>
<tr>
<td>17.6.1.2</td>
<td>Channel-specific machine data</td>
<td>1359</td>
</tr>
<tr>
<td>17.6.1.3</td>
<td>Axis/Spindle-specific machine data</td>
<td>1360</td>
</tr>
<tr>
<td>17.6.2</td>
<td>Setting data</td>
<td>1360</td>
</tr>
<tr>
<td>17.6.2.1</td>
<td>Channel-specific setting data</td>
<td>1360</td>
</tr>
<tr>
<td>17.6.2.2</td>
<td>Axis/spindle-specific setting data</td>
<td>1361</td>
</tr>
<tr>
<td>17.6.3</td>
<td>Signals</td>
<td>1361</td>
</tr>
<tr>
<td>17.6.3.1</td>
<td>Signals to channel</td>
<td>1361</td>
</tr>
<tr>
<td>17.6.3.2</td>
<td>Signals from channel</td>
<td>1361</td>
</tr>
<tr>
<td>17.6.3.3</td>
<td>Signals to axis/spindle</td>
<td>1361</td>
</tr>
<tr>
<td>17.6.3.4</td>
<td>Signals from axis/spindle</td>
<td>1362</td>
</tr>
<tr>
<td>18</td>
<td><strong>W1: Tool offset</strong></td>
<td>1363</td>
</tr>
<tr>
<td>18.1</td>
<td>Brief description</td>
<td>1363</td>
</tr>
<tr>
<td>18.2</td>
<td>Tool</td>
<td>1365</td>
</tr>
<tr>
<td>18.2.1</td>
<td>General</td>
<td>1365</td>
</tr>
<tr>
<td>18.2.2</td>
<td>Compensation memory structure</td>
<td>1368</td>
</tr>
<tr>
<td>18.2.3</td>
<td>Calculating the tool compensation</td>
<td>1370</td>
</tr>
<tr>
<td>18.2.4</td>
<td>Address extension for NC addresses T and M</td>
<td>1370</td>
</tr>
<tr>
<td>18.2.5</td>
<td>Free assignment of D numbers</td>
<td>1372</td>
</tr>
<tr>
<td>18.2.6</td>
<td>Compensation block in case of error during tool change</td>
<td>1378</td>
</tr>
<tr>
<td>18.2.7</td>
<td>Definition of the effect of the tool parameters</td>
<td>1381</td>
</tr>
<tr>
<td>18.3</td>
<td>Flat D number structure</td>
<td>1382</td>
</tr>
<tr>
<td>18.3.1</td>
<td>General</td>
<td>1382</td>
</tr>
<tr>
<td>18.3.2</td>
<td>Creating a new D number (compensation block)</td>
<td>1382</td>
</tr>
<tr>
<td>18.3.3</td>
<td>D number programming</td>
<td>1383</td>
</tr>
<tr>
<td>18.3.4</td>
<td>Programming the T number</td>
<td>1385</td>
</tr>
<tr>
<td>18.3.5</td>
<td>Programming M6</td>
<td>1385</td>
</tr>
<tr>
<td>18.3.6</td>
<td>Program test</td>
<td>1386</td>
</tr>
<tr>
<td>18.3.7</td>
<td>Tool management or &quot;Flat D numbers&quot;</td>
<td>1387</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>18.4</td>
<td>Tool cutting edge</td>
<td>1388</td>
</tr>
<tr>
<td>18.4.1</td>
<td>General</td>
<td>1388</td>
</tr>
<tr>
<td>18.4.2</td>
<td>Tool parameter 1: Tool type</td>
<td>1389</td>
</tr>
<tr>
<td>18.4.3</td>
<td>Tool parameter 2: Cutting edge position</td>
<td>1393</td>
</tr>
<tr>
<td>18.4.4</td>
<td>Tool parameters 3 - 5: Geometry - tool lengths</td>
<td>1395</td>
</tr>
<tr>
<td>18.4.5</td>
<td>Tool parameters 6 - 11: Geometry - tool shape</td>
<td>1396</td>
</tr>
<tr>
<td>18.4.6</td>
<td>Tool parameters 12 - 14: Wear - tool lengths</td>
<td>1398</td>
</tr>
<tr>
<td>18.4.7</td>
<td>Tool parameters 15 - 20: Wear - tool shape</td>
<td>1399</td>
</tr>
<tr>
<td>18.4.8</td>
<td>Tool parameters 21 - 23: Tool base dimension/adapter dimension</td>
<td>1400</td>
</tr>
<tr>
<td>18.4.9</td>
<td>Tool parameter 24: Undercut angle</td>
<td>1402</td>
</tr>
<tr>
<td>18.4.10</td>
<td>Tools with a relevant tool point direction</td>
<td></td>
</tr>
<tr>
<td>18.5</td>
<td>Tool radius compensation 2D (TRC)</td>
<td>1402</td>
</tr>
<tr>
<td>18.5.1</td>
<td>General</td>
<td>1402</td>
</tr>
<tr>
<td>18.5.2</td>
<td>Selecting the TRC (G41/G42)</td>
<td>1403</td>
</tr>
<tr>
<td>18.5.3</td>
<td>Approach and retraction behavior (NORM/KONT/KONTC/KONTT)</td>
<td>1404</td>
</tr>
<tr>
<td>18.5.4</td>
<td>Smooth approach and retraction</td>
<td>1409</td>
</tr>
<tr>
<td>18.5.4.1</td>
<td>Function</td>
<td>1409</td>
</tr>
<tr>
<td>18.5.4.2</td>
<td>Parameters</td>
<td>1410</td>
</tr>
<tr>
<td>18.5.4.3</td>
<td>Velocities</td>
<td>1417</td>
</tr>
<tr>
<td>18.5.4.4</td>
<td>System variables</td>
<td>1419</td>
</tr>
<tr>
<td>18.5.4.5</td>
<td>Supplementary conditions</td>
<td>1419</td>
</tr>
<tr>
<td>18.5.4.6</td>
<td>Examples</td>
<td>1420</td>
</tr>
<tr>
<td>18.5.5</td>
<td>Deselecting the TRC (G40)</td>
<td>1423</td>
</tr>
<tr>
<td>18.5.6</td>
<td>Compensation at outside corners</td>
<td>1424</td>
</tr>
<tr>
<td>18.5.7</td>
<td>Compensation and inner corners</td>
<td>1428</td>
</tr>
<tr>
<td>18.5.8</td>
<td>Collision detection and bottleneck detection</td>
<td>1430</td>
</tr>
<tr>
<td>18.5.9</td>
<td>Blocks with variable compensation value</td>
<td>1432</td>
</tr>
<tr>
<td>18.5.10</td>
<td>Keep tool radius compensation constant</td>
<td>1434</td>
</tr>
<tr>
<td>18.5.11</td>
<td>Alarm behavior</td>
<td>1437</td>
</tr>
<tr>
<td>18.5.12</td>
<td>Intersection procedure for polynomials</td>
<td>1437</td>
</tr>
<tr>
<td>18.5.13</td>
<td>G461/G462 Approach/retract strategy expansion</td>
<td>1438</td>
</tr>
<tr>
<td>18.6</td>
<td>Toolholder with orientation capability</td>
<td>1442</td>
</tr>
<tr>
<td>18.6.1</td>
<td>General</td>
<td>1442</td>
</tr>
<tr>
<td>18.6.2</td>
<td>Kinematic interaction and machine design</td>
<td>1449</td>
</tr>
<tr>
<td>18.6.3</td>
<td>Inclined surface machining with 3 + 2 axes</td>
<td>1457</td>
</tr>
<tr>
<td>18.6.4</td>
<td>Machine with rotary work table</td>
<td>1458</td>
</tr>
<tr>
<td>18.6.5</td>
<td>Procedure when using toolholders with orientation capability</td>
<td>1461</td>
</tr>
<tr>
<td>18.6.6</td>
<td>Programming</td>
<td>1465</td>
</tr>
<tr>
<td>18.6.7</td>
<td>Supplementary conditions and control system response for orientation</td>
<td>1466</td>
</tr>
<tr>
<td>18.7</td>
<td>Cutting edge data modification for tools that can be rotated</td>
<td>1468</td>
</tr>
<tr>
<td>18.7.1</td>
<td>Function</td>
<td>1468</td>
</tr>
<tr>
<td>18.7.2</td>
<td>Determination of angle of rotation</td>
<td>1468</td>
</tr>
<tr>
<td>18.7.3</td>
<td>Cutting edge position, cut direction and angle for rotary tools</td>
<td>1469</td>
</tr>
<tr>
<td>18.7.4</td>
<td>Modifications during the rotation of turning tools</td>
<td>1472</td>
</tr>
<tr>
<td>18.7.5</td>
<td>Cutting edge position for milling and tapping tools</td>
<td>1474</td>
</tr>
<tr>
<td>18.7.6</td>
<td>Modifications during rotation of milling and tapping tools</td>
<td>1475</td>
</tr>
<tr>
<td>18.7.7</td>
<td>Parameter assignment</td>
<td>1476</td>
</tr>
<tr>
<td>18.7.8</td>
<td>Programming</td>
<td>1477</td>
</tr>
<tr>
<td>18.7.9</td>
<td>Example</td>
<td>1480</td>
</tr>
<tr>
<td>18.8</td>
<td>Incrementally programmed compensation values</td>
<td>1481</td>
</tr>
</tbody>
</table>
### Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.8.1</td>
<td>G91 extension</td>
<td>1481</td>
</tr>
<tr>
<td>18.8.2</td>
<td>Machining in direction of tool orientation</td>
<td>1483</td>
</tr>
<tr>
<td>18.9</td>
<td>Basic tool orientation</td>
<td>1484</td>
</tr>
<tr>
<td>18.10</td>
<td>Special handling of tool compensations</td>
<td>1488</td>
</tr>
<tr>
<td>18.10.1</td>
<td>Relevant setting data</td>
<td>1488</td>
</tr>
<tr>
<td>18.10.2</td>
<td>Mirror tool lengths (SD42900 $SC_MIRROR_TOOL_LENGTH)</td>
<td>1489</td>
</tr>
<tr>
<td>18.10.3</td>
<td>Mirror wear lengths (SD42920 $SC_WEAR_SIGN_CUTPOS)</td>
<td>1490</td>
</tr>
<tr>
<td>18.10.4</td>
<td>Tool length and plane change (SD42940 $SC_TOOL_LENGTH_CONST)</td>
<td>1491</td>
</tr>
<tr>
<td>18.10.5</td>
<td>Tool type (SD42950 $SC_TOOL_LENGTH_TYPE)</td>
<td>1492</td>
</tr>
<tr>
<td>18.10.6</td>
<td>Tool lengths in the WCS, allowing for the orientation</td>
<td>1493</td>
</tr>
<tr>
<td>18.10.7</td>
<td>Tool length offsets in tool direction</td>
<td>1493</td>
</tr>
<tr>
<td>18.11</td>
<td>Sum offsets and setup offsets</td>
<td>1498</td>
</tr>
<tr>
<td>18.11.1</td>
<td>General</td>
<td>1498</td>
</tr>
<tr>
<td>18.11.2</td>
<td>Description of function</td>
<td>1499</td>
</tr>
<tr>
<td>18.11.3</td>
<td>Activation</td>
<td>1502</td>
</tr>
<tr>
<td>18.11.4</td>
<td>Examples</td>
<td>1507</td>
</tr>
<tr>
<td>18.11.5</td>
<td>Upgrades for Tool Length Determination</td>
<td>1508</td>
</tr>
<tr>
<td>18.11.5.1</td>
<td>Taking the compensation values into account location-specifically and workpiece-specifically</td>
<td>1508</td>
</tr>
<tr>
<td>18.11.5.2</td>
<td>Functionality of the individual wear values</td>
<td>1512</td>
</tr>
<tr>
<td>18.12</td>
<td>Working with tool environments</td>
<td>1516</td>
</tr>
<tr>
<td>18.12.1</td>
<td>General</td>
<td>1516</td>
</tr>
<tr>
<td>18.12.2</td>
<td>Saving with TOOLENV</td>
<td>1516</td>
</tr>
<tr>
<td>18.12.3</td>
<td>Delete tool environment</td>
<td>1518</td>
</tr>
<tr>
<td>18.12.4</td>
<td>How many environments and which ones are saved?</td>
<td>1519</td>
</tr>
<tr>
<td>18.12.5</td>
<td>Read T. D. DL from a tool environment</td>
<td>1520</td>
</tr>
<tr>
<td>18.12.6</td>
<td>Read tool lengths, tool length components</td>
<td>1521</td>
</tr>
<tr>
<td>18.12.7</td>
<td>Changing tool components</td>
<td>1526</td>
</tr>
<tr>
<td>18.13</td>
<td>Tool lengths L1, L2, L3 assignment: LENTOAX</td>
<td>1531</td>
</tr>
<tr>
<td>18.14</td>
<td>Supplementary conditions</td>
<td>1534</td>
</tr>
<tr>
<td>18.14.1</td>
<td>Flat D number structure</td>
<td>1534</td>
</tr>
<tr>
<td>18.14.2</td>
<td>SD42935 expansions</td>
<td>1535</td>
</tr>
<tr>
<td>18.15</td>
<td>Examples</td>
<td>1535</td>
</tr>
<tr>
<td>18.15.1</td>
<td>Toolholder with orientation capability</td>
<td>1535</td>
</tr>
<tr>
<td>18.15.1.1</td>
<td>Example: Toolholder with orientation capability</td>
<td>1535</td>
</tr>
<tr>
<td>18.15.1.2</td>
<td>Example of toolholder with orientation capability with rotary table</td>
<td>1536</td>
</tr>
<tr>
<td>18.15.1.3</td>
<td>Basic tool orientation example</td>
<td>1538</td>
</tr>
<tr>
<td>18.15.1.4</td>
<td>Calculation of compensation values on a location-specific and workpiece-specific basis</td>
<td>1539</td>
</tr>
<tr>
<td>18.15.2</td>
<td>Examples 3-6: SETTCOR function for tool environments</td>
<td>1541</td>
</tr>
<tr>
<td>18.16</td>
<td>Data lists</td>
<td>1547</td>
</tr>
<tr>
<td>18.16.1</td>
<td>Machine data</td>
<td>1547</td>
</tr>
<tr>
<td>18.16.1.1</td>
<td>NC-specific machine data</td>
<td>1547</td>
</tr>
<tr>
<td>18.16.1.2</td>
<td>Channelspecific machine data</td>
<td>1548</td>
</tr>
<tr>
<td>18.16.1.3</td>
<td>Axis/spindlespecific machine data</td>
<td>1549</td>
</tr>
<tr>
<td>18.16.2</td>
<td>Setting data</td>
<td>1549</td>
</tr>
<tr>
<td>18.16.2.1</td>
<td>Channelspecific setting data</td>
<td>1549</td>
</tr>
<tr>
<td>18.16.3</td>
<td>Signals</td>
<td>1550</td>
</tr>
<tr>
<td>18.16.3.1</td>
<td>Signals from channel</td>
<td>1550</td>
</tr>
</tbody>
</table>
Z1: NC/PLC interface signals

19.1 Various interface signals and functions (A2)........................................................................1551
  19.1.1 Signals from PLC to NC (DB10).....................................................................................1551
  19.1.2 Selection/Status signals from HMI to PLC (DB10)........................................................1551
  19.1.3 Signals from the NC to the PLC (DB10)........................................................................1552
  19.1.4 Signals to Operator Panel (DB19)................................................................................1556
  19.1.5 Signals from operator control panel (DB19).................................................................1561
  19.1.6 Signals to channel (DB21, ...).....................................................................................1565
  19.1.7 Signals from channel (DB21, ...)..................................................................................1565
  19.1.8 Signals to axis/spindle (DB31, ...)................................................................................1566
  19.1.9 Signals from axis/spindle (DB31, ...).............................................................................1578

19.2 Axis monitoring, protection zones (A3)..............................................................................1587
  19.2.1 Signals to channel (DB21, ...).....................................................................................1587
  19.2.2 Signals from channel (DB21, ...)..................................................................................1588
  19.2.3 Signals to axis/spindle (DB31, ...)................................................................................1589
  19.2.4 Signals from axis/spindle (DB31, ...)..........................................................................1591

19.3 Continuous-path mode, exact stop and LookAhead (B1)..................................................1592
  19.3.1 Signals from channel (DB21, ...)..................................................................................1592
  19.3.2 Signals from axis/spindle (DB31, ...).............................................................................1593

19.4 Travel to fixed stop (F1)......................................................................................................1594
  19.4.1 Signals to axis/spindle (DB31, ...)................................................................................1594
  19.4.2 Signals from axis/spindle (DB31, ...)..........................................................................1595

19.5 Help function output to PLC (H2).....................................................................................1596
  19.5.1 Signals to channel (DB21, ...).....................................................................................1596
  19.5.2 Signals from channel (DB21, ...)..................................................................................1596
  19.5.3 Signals from axis/spindle (DB31, ...)..........................................................................1600

19.6 Mode group, channel, program operation, reset response (K1)..........................................1601
  19.6.1 Signals to mode group (DB11)....................................................................................1601
  19.6.2 Signals from the mode group (DB11)...........................................................................1605
  19.6.3 Signals to channel (DB21, ...).....................................................................................1609
  19.6.4 Signals from channel (DB21, ...)..................................................................................1613
  19.6.5 Signals to axis/spindle (DB31, ...)................................................................................1627
  19.6.6 Signals from axis/spindle (DB31, ...)..........................................................................1627

19.7 Axis types, coordinate systems, frames (K2)......................................................................1629
  19.7.1 Signals to axis/spindle (DB31, ...)................................................................................1629

19.8 Emergency stop (N2).........................................................................................................1629
  19.8.1 Signals to NC (DB10)...................................................................................................1629
  19.8.2 Signals from NC (DB10)..............................................................................................1631

19.9 PLC basic program (P3)......................................................................................................1631

19.10 Reference point approach (R1).......................................................................................1632
  19.10.1 Signals to channel (DB21, ...)....................................................................................1632
  19.10.2 Signals from channel (DB21, ...)................................................................................1633
  19.10.3 Signals to axis/spindle (DB31, ...)..............................................................................1633
  19.10.4 Signals from axis/spindle (DB31, ...)..........................................................................1634

19.11 Spindles (S1)..................................................................................................................1636
  19.11.1 Signals to axis/spindle (DB31, ...)..............................................................................1636
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.11.2</td>
<td>Signals from axis/spindle (DB31, ...)</td>
<td>1642</td>
</tr>
<tr>
<td>19.12</td>
<td>Feeds (V1)</td>
<td>1652</td>
</tr>
<tr>
<td>19.12.1</td>
<td>Signals to channel (DB21, ...)</td>
<td>1652</td>
</tr>
<tr>
<td>19.12.2</td>
<td>Signals to axis/spindle (DB31, ...)</td>
<td>1663</td>
</tr>
<tr>
<td>19.12.3</td>
<td>Signals from axis/spindle (DB31, ...)</td>
<td>1670</td>
</tr>
<tr>
<td><strong>A</strong></td>
<td>Appendix</td>
<td></td>
</tr>
<tr>
<td>A.1</td>
<td>List of abbreviations</td>
<td>1671</td>
</tr>
<tr>
<td>A.2</td>
<td>Documentation overview</td>
<td>1680</td>
</tr>
<tr>
<td><strong>Glossary</strong></td>
<td></td>
<td>1681</td>
</tr>
<tr>
<td><strong>Index</strong></td>
<td></td>
<td>1703</td>
</tr>
</tbody>
</table>
1

1.1 General safety instructions

<table>
<thead>
<tr>
<th>WARNING</th>
<th>Risk of death if the safety instructions and remaining risks are not carefully observed</th>
</tr>
</thead>
<tbody>
<tr>
<td>WARNING</td>
<td>If the safety instructions and residual risks are not observed in the associated hardware documentation, accidents involving severe injuries or death can occur.</td>
</tr>
<tr>
<td></td>
<td>- Observe the safety instructions given in the hardware documentation.</td>
</tr>
<tr>
<td></td>
<td>- Consider the residual risks for the risk evaluation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WARNING</th>
<th>Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.</td>
</tr>
<tr>
<td></td>
<td>- Protect the parameterization (parameter assignments) against unauthorized access.</td>
</tr>
<tr>
<td></td>
<td>- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).</td>
</tr>
</tbody>
</table>

1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit Hotspot-Text (http://www.siemens.com/industrialsecurity).

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit Hotspot-Text (http://support.automation.siemens.com).
WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can result in death, severe injuries and/or material damage.

- Keep the software up to date.
  You will find relevant information and newsletters at this address (http://support.automation.siemens.com).

- Incorporate the automation and drive components into a holistic, state-of-the-art industrial security concept for the installation or machine.
  You will find further information at this address (http://www.siemens.com/industrialsecurity).

- Make sure that you include all installed products into the holistic industrial security concept.
2.1 Brief description

Contents

The PLC/NCK interface comprises a data interface on one side and a function interface on the other. The data interface contains status and control signals, auxiliary functions and G functions, while the function interface is used to transfer jobs from the PLC to the NCK.

This Description describes the functionality of interface signals, which are of general relevance but are not included in the Descriptions of Functions.

● Asynchronous events
● Status signals
● PLC variable (read and write)

2.2 NC/PLC interface signals - only 840D sl

2.2.1 General

NC/PLC interface

The NC/PLC interface comprises the following parts:

● Data interface
● Function interface

Data interface

The data interface is used for component coordination:

● PLC user program
● NC
● HMI (operator control component)
● MCP (machine control panel)

Data exchange is organized by the basic PLC program.
Cyclic signal exchange

The following interface signals are transferred cyclically, i.e. in the clock grid of the OB1, by the basic PLC program:

- NC and operator-panel-front-specific signals
- Mode group-specific signals
- Channel-specific signals
- Axis/spindle-specific signals

NC and operator-panel-front-specific signals (DB10)

PLC to NC:
- Signals for influencing the CNC inputs and outputs
- Keyswitch signals (and password)

NC to PLC:
- Actual values of CNC inputs
- Setpoints of CNC outputs
- Ready signals from NC and HMI
- NC status signals (alarm signals)

Channel-specific signals (DB21, ...)

PLC to NC:
- Control signal "Delete distance-to-go"

NC to PLC:
- NC status signals (NCK alarm active)

Axis/spindle-specific signals (DB31, etc.)

PLC to NC:
- Control signals to axis/spindle (e.g. follow-up mode, servo enable, etc.)
- Control signals to drive (bytes 20, 21)

NC to PLC:
- Status signals from axis/spindle (e.g. position controller active, current controller active, etc.)
- Control signals from drive (bytes 93, 94)

Function interface

The function interface is generated by function blocks (FB) and function calls (FC). Function requests, e.g. to traverse axes, are sent from the PLC to the NC via the function interface.
References

- Description of the basic PLC program:
  → Function Manual, Basic Functions, Basic PLC Program (P3)
- Description of the event-driven signal exchange (auxiliary and G functions):
  → Function Manual, Basic Functions; Auxiliary Function Output to PLC (H2)
- Overview of all interface signals, functional and data components:
  → List Manual 2

2.2.2 Ready signal to PLC

**DB10 DBX104.7 (NCK CPU ready)**

The NCK CPU is ready and registers itself cyclically with the PLC.

**DB10 DBX108.3 (HMI - CPU at OPI ready)**

SINUMERIK Operate is ready and registers itself cyclically with the NC.

**DB10 DBX108.5 (drives in cyclic operation)**

Requirement: For all machine axes of the NC, the associated drives are in the cyclic operation, i.e. they cyclically exchange PROFIdrive telegrams with the NC.

**DB10 DBX108.6 (drive ready)**

Requirement: For all machine axes of the NC, the associated drives and also the third-party drives are ready via PROFIBUS:

DB31, ... DBX93.5 == 1 (drive ready)

**DB10 DBX108.7 (NC ready)**

The NC is ready.

2.2.3 Status signals to PLC

**DB10 DBX103.0 (remote diagnosis active)**

The HMI component reports to the PLC that the remote diagnostics (option) is active, i.e. controlling is done via an external PC.
DB10 DBX109.6 (ambient temperature alarm)

The ambient temperature or fan monitoring function has responded.

DB10 DBX109.7 (NCK battery alarm)

The battery voltage has dropped below the lower limit value. The control can still be operated. A control system shutdown or failure of the supply voltage will lead to loss of data.

DB10 DBX109.0 (NCK alarm pending)

The NC signals that at least one NC alarm is pending. The channel-specific interface can be scanned to see which channels are involved and whether this will cause a processing stop.

DB21, ... DBX36.6 (channel-specific NCK alarm pending)

The NC sends this signal to the PLC to indicate that at least one NC alarm is pending for the relevant channel (see also DB21, ... DBX36.7).

DB21, ... DBX36.7 (NCK alarm with processing stop present)

The NC sends this signal to the PLC to indicate that at least one NCK alarm which has interrupted or aborted the current program run (processing stop) is pending for the affected channel.

2.2.4 Signals to/from the operator panel front

DB19 DBX0.0 (screen bright)

The screen blanking is disabled.

DB19 DBX0.1 (screen dark)

The operator panel screen is darkened.

If the interface signal is used to actively darken the screen:

- It is no longer possible to switch the screen bright again on the keyboard (see below).
- The first keystroke on the operator panel front already triggers an operator action.

**Note**

In order to prevent accidental operator actions when the screen is darkened via the interface signal, we recommend disabling the keyboard at the same time.

DB19 DBX0.1 = 1 AND DB19 DBX0.2 = 1 (key disable)
Screen darkening via keyboard/automatic screen saver

If no buttons are pressed on the operator panel front within the assigned time (default = 3 minutes):

MD9006 $MM_DISPLAY_BLACK_TIME
(time for screen darkening), the screen is automatically darkened.

The screen lights up again the first time a button is pressed following darkening. Pressing a
button to lighten the screen will not generate an operator action.

Parameterization:

DB19 DBX0.1 = 0

MD9006 $MM_DISPLAY_BLACK_TIME > 0

DB19 DBX0.2 (key disable)

All inputs via the connected keyboard are inhibited.

DB19 DBX 0.3 / 0.4 (Delete cancel alarms / Delete recall alarms) (HMI Advanced)

Request to delete all currently pending alarms with Cancel or Recall delete criterion. Deletion
of the alarms is acknowledged via the following interface signals.

- DB19 DBX20.3 (cancel alarm deleted)
- DB19 DBX20.4 (recall alarm deleted)

DB19 DBX0.7 (actual values in WCS, 0=MCS)

Switching over of actual-value display between machine and workpiece coordinate system:

- DB19 DBX0.7 = 0: Machine coordinate system (MCS)
- DB19 DBX0.7 = 1: Workpiece coordinate system (WCS)

DB19 DBB13 (control of the file transfer via hard disk) (HMI Advanced only)

Job byte to control file transfer via hard disk. The jobs relate to the user control file in the
interface signals:

DB19 DBB16 (part program handling: Number of the control file for user file names)
DB19 DBB17 (part program handling: Index of the file to be transmitted from the user list)

DB19 DBB16 (control of file transfer via hard disk) (HMI Advanced only)

Control byte for file transfer via hard disk to define the index for the control file (job list). This
file is handled according to the job in the interface signal:

DB19 DBB13
DB19 DBB17 (part program handling: Index of the file to be transferred from the user list)

Control byte for file transfer via hard disk to indicate the line in the user control file in which the control file to be transferred is stored.

DB19 DBB26 (part program handling: Status)

Status byte for current status of data transfer for "select", "load" or "unload", or if an error occurred during data transmission.

DB19 DBB27 (error program handling)

Output byte for error values for data transfer via hard disk.

2.2.5 Signals to channel

DB21, ... DBX6.2 (delete distance-to-go)

The rising edge on the interface signal generates a stop on the programmed path in the corresponding NC channel with the currently active path acceleration. The path distance-to-go is then deleted and the block change to the next part-program block is enabled.

2.2.6 Signals to axis/spindle

DB31, ... DBX1.0 (drive test travel enable)

<table>
<thead>
<tr>
<th>NOTICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifications for the drive test</td>
</tr>
<tr>
<td>It is the sole responsibility of the machine manufacturer / system startup engineer to take suitable action / carry out appropriate tests to ensure that the machine axis can be traversed during the drive test without putting personnel or machinery at risk.</td>
</tr>
</tbody>
</table>

If machine axes are traversed by special test functions such as "function generator", an explicit drive-test-specific enable is requested for the motion:

DB31, ... DBX61.0 = 1 (drive test travel request)

The motion is carried out once the motion is enabled:

DB31, ... DBX1.0 == 1 (drive test travel enable)
DB31, ... DBX1.3 (axis/spindle disable)

**Stationary axis**
If the interface signal is set for a stationary axis, all travel request are ignored as of this time.

The travel requests are retained. If the axis disable is cancelled when a traversing request is pending DB31, ... DBX1.3 = 0 the motion is carried out.

**Traversing axis**
If the interface signal is set for a traversing axis, the axis is stopped using the currently active brake characteristic. If the interface signal is reset while the travel request is still present, traversing is continued.

**Spindle**
- Open-loop control mode: Speed setpoint zero is output immediately
- Positioning mode: See "Stationary axis" / "Traversing axis" above

DB31, ... DBX1.4 (follow-up mode)

The interface signal is only effective with the DB31, ... DBX2.1 (controller enable) interface signal

<table>
<thead>
<tr>
<th>DB31, ... DBX2.1</th>
<th>DB31, ... DBX1.4</th>
<th>Operating mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Ineffective, In closed-loop control</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Follow-up</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Hold</td>
</tr>
</tbody>
</table>

**Follow-up**
In follow-up mode, the set position of the machine axis is continuously corrected to the actual position (set position = actual position).

Feedback: DB31, ... DBX61.3 == 1 (follow-up active)

During follow-up mode, clamping or standstill monitoring are not active.

**Note**
If the DB31, ... DBX2.1 (controller enable) interface signal is set during follow-up mode, repositioning to the last programmed position is performed with REPOSA (approach on a straight line with all axes) when the NC program is active. In all other cases, all subsequent motions start at the actual position.

**Hold**
In "Hold" mode, the set position of the machine axis does not follow the actual position. If the machine axis is moved away from the set position, the difference between the set position and the actual position increases constantly (following error). The following error is suddenly corrected if the controller enable is set without complying with the axial acceleration characteristic (speed jump).

Feedback: DB31, ... DBX61.3 == 0 (follow-up active)
During "hold", clamping or standstill monitoring are active.

**Note**
The following error is suddenly corrected if the controller enable is set without complying with the axial acceleration characteristic (speed jump).

**Application example**
Positioning response of machine axis Y following clamping when "controller enable" is set. Clamping pushed the machine axis from the actual position $Y_1$ to the clamping position $Y_k$.

![Figure 2-1 Effect of controller enable and follow-up mode](image)

1. IS "Follow-up mode" = 0
2. Remove IS "Servo enable"
3. Terminals
4. Disconnect terminals
5. Set IS "Servo enable"

![Figure 2-2 Trajectory for clamping and "hold"](image)
Drives with analog setpoint interface

A drive with an analog setpoint interface is capable of traversing the machine axis with an external setpoint. If "follow-up mode" is set for the machine axis, the actual position continues to be acquired. Once follow-up mode has been cancelled, referencing is not required.

The following procedure is recommended:

1. Activate follow-up mode:
   - DB31, ... DBX2.1 = 0 (controller enable)
   - DB31, ... DBX1.4 = 1 (follow-up mode) (in the same or preceding OB1 cycle)
   → The axis/spindle is operating in follow-up mode

2. Deactivate external controller enable and external speed setpoint
   → Axis/spindle moves with external setpoint
   → NC continues to detect the actual position and corrects the set position to the actual position

3. Deactivate external controller enable and cancel external speed setpoint
   → Axis/spindle stops

4. Cancelling follow-up mode
   - DB31, ... DBX2.1 = 1 (controller enable)
   - DB31, ... DBX1.4 = 0 (follow-up mode)
   → NC synchronizes to actual position. The next traversing motion begins at this position.

**Note**

"Follow-up mode" does not have to be cancelled because it only has an effect in combination with "controller enable".

Cancelling follow-up mode

Once follow-up mode has been cancelled, the machine axis does not have to be referenced again if the maximum permissible encoder limit frequency of the active measuring system was
not exceeded during follow-up mode. If the encoder limit frequency is exceeded, the controller will detect this:

- DB31, ... DBX60.4 / 60.5 = 0 (referenced/synchronized 1 / 2)
- Alarm: "21610 Encoder frequency exceeded"

**Note**

If "follow-up mode" is deactivated for a machine axis, which is part of an active transformation (e.g. TRANSMIT), this can generate motions as part of repositioning (REPOS) other machine axes involved in the transformation.

**Monitoring**

If a machine axis is in follow-up mode, the following monitoring mechanisms will not act:

- Standstill monitoring
- Clamping monitoring
- Positioning monitoring

**Effects on other interface signals:**

- DB31, ... DBX60.7 = 0 (position reached with exact stop fine)
- DB31, ... DBX60.6 = 0 (position reached with exact stop coarse)

**DB31, ... DBX1.5/1.6 (position measuring system 1/2)**

Two measuring systems can be connected to one machine axis, e.g.

- Indirect motor measuring system
- Direct measuring system on load

Only one measuring system can be active at any one time. All closed-loop control, positioning operations, etc. involving the machine axis always relate to the active measuring system.

---

![Diagram showing position measuring systems 1 and 2](image-url)

**Figure 2-4** Position measuring systems 1 and 2
Functionality of the "Position measuring system 1 / 2" interface signals in conjunction with the "Controller enable":

<table>
<thead>
<tr>
<th>DB31, ... DBX1.5</th>
<th>DB31, ... DBX1.6</th>
<th>DB31, ... DBX2.1</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Any</td>
<td>1</td>
<td>Position measuring system 1 active</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Position measuring system 2 active</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>&quot;Parking&quot; active</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Spindle without position measuring system (speed-controlled)</td>
</tr>
<tr>
<td>1 → 0</td>
<td>0 → 1</td>
<td>1</td>
<td>Switchover: Position measuring system 1 → 2</td>
</tr>
<tr>
<td>0 → 1</td>
<td>1 → 0</td>
<td>1</td>
<td>Switchover: Position measuring system 2 → 1</td>
</tr>
</tbody>
</table>

**DB31, ... DBX2.1 (controller enable)**

Setting the controller enable closes the machine axis position control loop. The machine axis is in position control mode.

DB31, ... DBX2.1 == 1

Cancelling the controller enable opens the machine axis position control loop and, subject to a delay, the machine axis speed control loop:

DB31, ... DBX2.1 == 0

**Activation methods**

The closed-loop controller enable for a machine axis is influenced by:

- **NC/PLC interface signal:**
  - DB31, ... DBX2.1 (controller enable)
  - DB31, ... DBX21.7 (pulse enable)
  - DB31, ... DBX93.5 (drive ready)
  - DB10, DBX56.1 (emergency stop)

- **NCK-internal**
  Alarms that trigger cancellation of the controller enable on the machine axes. The alarm responses are described in:

  **References:**
  Diagnostics Manual

**Cancelling the controller enable when the machine axis is at standstill:**

- The machine axis position control loop opens
- DB31, ... DBX61.5 == 0 (position controller active)

**Cancelling the controller enable when the machine axis is in motion:**

If a machine axis is part of an interpolatory path motion or coupling and the controller enable for this is cancelled, all axes involved are stopped with a fast stop (speed setpoint = 0) and an alarm is displayed:
Alarm: "21612 Controller enable reset during motion"

- The machine axis is decelerated taking into account the parameterized duration of the braking ramp for error states with a fast stop (speed setpoint = 0):
  MD36610 $MA_AX_EMERGENCY_STOP_TIME (max. duration of the braking ramp in the event of errors)
  An alarm is displayed:
  Alarm: "21612 Controller enable reset during motion"

Note
The controller enable is cancelled at the latest when the cut-off delay expires:
MD36610 $MA_AX_EMERGENCY_STOP_TIME

- The machine axis position control loop opens. Feedback via interface signal:
  DB31, ... DBX61.5 == 0 (position controller active).
  The time for the parameterized cut-off delay of the controller enable is started by the machine data:
  MD36620 $MA_SERVO_DISABLE_DELAY_TIME (OFF delay of the controller enable)

- As soon as the actual speed has reached the zero speed range, the drive controller enable is cancelled. Feedback via interface signal:
  DB31, ... DBX61.6 == 0 (speed controller active)

- The actual position value of the machine axis continues to be acquired by the controller.

- At the end of the braking operation, the machine axis is switched to follow-up mode, regardless of the corresponding NC/PLC interface signal. Standstill and clamping monitoring are not effective. See the description above for the interface signal:
  DB31, ... DBX1.4 (follow-up mode).

Synchronizing the actual value (reference point approach)

Once the controller enable has been set, the actual position of the machine axis does not need to be synchronized again (reference point approach) if the maximum permissible limit frequency of the measuring system was not exceeded during the time in which the machine axis was not in position-control mode.
**DB31, ..., DBX2.2 (distance-to-go/spindle reset (axis/spindle-specific))**

"Delete distance-to-go" is effective in AUTOMATIC and MDA modes only in conjunction with positioning axes. The positioning axis is decelerated to standstill following the current brake characteristic. The distance-to-go of the axis is deleted.

**Spindle reset**

For a detailed description of the spindle reset, see Section "S1: Spindles (Page 1199)".

**DB31, ..., DBX9.0 / 9.1 / 9.2 (controller parameter set)**

Request for activation of the specified controller parameter set.

<table>
<thead>
<tr>
<th>Controller parameter set</th>
<th>DBX9.2</th>
<th>DBX9.1</th>
<th>DBX9.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Parameter set changeover must be enabled via the machine data (not required for spindles):

\[ \text{MD35590 SMA\_PARAMSET\_CHANGE\_ENABLE = 1 or 2} \]

For detailed information on the parameter set changeover, see Section "Parameter set selection during gear step change (Page 1246)".

### Parameter set changeover when machine axis is in motion

The response to a parameter set changeover depends on the consequential change in the closed-loop control circuit gain factor \( K_v \):

- "Identical servo gain factors (\( K_v \))" or "position control not active":
  - The NC responds immediately to the parameter set changeover. The parameter set is also changed during the motion.

- "Unequal servo gain factors (\( K_v \))" or "position control not active":
  - In order to effect a changeover as smoothly as possible, changeover is not activated until the axis "is stationary", i.e. once the parameterized zero speed has been reached or undershot:
    - \( \text{DB31, ... DBX61.4 = 1 (axis/spindle stationary)} \)
    - \( \text{MD36060 SMA\_STANDSTILL\_VELO\_TOL (threshold velocity/speed "axis/spindle stationary")} \)

### Parameter set changeover from the part program

For parameter set changeover from the part program, the user (machine manufacturer) must define corresponding user-specific auxiliary functions and evaluate them in the PLC user program. The PLC user program will then set the changeover request on the corresponding parameter set.

For detailed information on the auxiliary function output, see Section "H2: Auxiliary function outputs to PLC (Page 377)".

#### DB31, ... DBX9.3 (parameter set specification disabled by NC)

If the interface signal is set, a request for a parameter set changeover (DB31, ... DBX9.0 / 9.1 / 9.2) is ignored.

### 2.2.7 Signals from axis/spindle

#### DB31, ... DBX61.0 (drive test travel request)

If machine axes are traversed by special test functions such as "function generator", an explicit drive-test-specific enable is requested for the movement:
DB31, ... DBX61.0 == 1 (drive test travel request)
The motion is carried out once the motion is enabled:
DB31, ... DBX1.0 == 1 (drive test travel enable)

DB31, ... DBX61.3 (follow-up active)
The machine axis is in follow-up mode.

DB31, ... DBX61.4 (axis/spindle stationary \( n < n_{\text{min}} \))
"Axis/spindle stationary" is set by the NC if:
- No new setpoints are to be output \( \text{AND} \)
- The actual speed of the machine axis is lower than the parameterized zero speed:
  MD36060 $\text{MA_STANDSTILL_VELO_TOL}$ (threshold velocity axis stationary)

DB31, ... DBX61.5 (position controller active)
The machine axis position control loop is closed and position control is active.

DB31, ... DBX61.6 (speed controller active)
The machine axis speed control loop is closed and speed control is active.

DB31, ... DBX61.7 (current controller active)
The machine axis current control loop is closed and current control is active.

DB31, ... DBX69.0 / 69.1 / 69.2 (parameter set servo)
Active parameter set Coding accordingly:
DB31, ... DBX9.0 / 9.1 / 9.2 (controller parameter set selection)

DB31, ... DBX76.0 (lubrication pulse)
Following a control POWER ON/RESET, the signal status is 0 (FALSE).
The "lubrication pulse" is inverted (edge change), as soon as the machine axis has covered
the parameterized traversing distance for lubrication:
MD33050 $\text{MA_LUBRICATION_DIST}$ (distance for lubrication by PLC)
2.2.8 Signals to axis/spindle (digital drives)

DB31, ... DBX21.0 - 4 (requesting a switchover of a motor and/or drive data set)

The PLC issues a request to the drive to switch over to a new motor and/or drive data set.

The interface can be flexibly parameterized using: DB31, ... DBX130.0 - 4 (see Chapter "Switching over motor/drive data sets (Page 65)"

The active motor and/or drive data set is indicated using: DB31, ... DBX93.0 - 4

Note

Main spindle drive

For a main spindle drive, only motor data sets 1 and 2 are valid when switching over between a star and delta connection.

- Motor data set 1: Star
- Motor data set 2: Delta

DB31, ... DBX21.5 (Motor selection done)

The PLC user program sends this signal to the drive to indicate successful motor selection. The pulses are then enabled by the drive.

DB31, ... DBX21.6 (integrator disable, n-controller)

The PLC user program inhibits the integrator of the speed controller for the drive. The speed controller is switched over from a PI to P controller.

If the speed controller integrator disable is activated, compensations might take place in certain applications (e.g. if the integrator was already holding a load while stationary).

Feedback signal via: DB31, ... DBX93.6 = 1 (integrator n-controller disabled)

DB31, ... DBX21.7 (request pulse enable)

The pulse enable for the drive module is only requested if all enable signals (hardware and software) are pending:

- Trigger equipment enable
- Controller and pulse enable
- Pulse enable (safe operating stop)
- Stored hardware input
● Setpoint enable
● "Ready to run state"
  – No drive alarm (DClink1 error)
  – DC link connected
  – Ramp-up completed
Feedback signal via: DB31, ... DBX93.7 (pulses enabled)

2.2.9 Signals from axis/spindle (digital drives)

DB31, ... DBX92.1 (ramp-function generator disable active)
The drive signals back to the PLC that ramp-function-generator fast stop is active. The drive is thus brought to a standstill without the ramp function (with speed setpoint 0).

DB31, ... DBX93.0 - 4 (indicates active motor and drive data set)
The drive signals back the active motor and drive data set to the PLC.
The interface can be flexibly parameterized using: DB31, ... DBX130.0 - 4 (see Chapter "Switching over motor/drive data sets (Page 65)"
The request to switch over motor and/or drive data set is realized using: DB31, ... DBX21.0 - 4

DB31, ... DBX93.5 (DRIVE ready)
Checkback signal indicating that the drive is ready. The conditions required for traversing the axis/spindle are fulfilled.

DB31, ... DBX93.6 (integrator n-controller disabled)
The speed-controller integrator is disabled. The speed controller has thus been switched from PI to P controller.

DB31, ... DBX93.7 (pulses enabled)
The pulse enable for the drive module is available. The axis/spindle can now be traversed.

DB31, ... DBX94.0 (motor temperature prewarning)
The temperature of the motor is higher than the set motor temperature warning threshold (drive parameter p0604).
See also note below for "DB31, ... DBX94.1 (heat sink temperature prewarning)".
DB31, ... DBX94.1 (heat sink temperature prewarning)

The temperature of the heat sink in the power unit is outside the permissible range. If the overtemperature remains, the drive switches itself off after approx. 20 s.

Note

Temperature prewarning DB31, ... DBX94.0 and DBX94.1

The interface signals are derived from the following signals of the cyclic drive telegram:

- Case 1: Temperature warning in the message word
  - DB31, ... DBX94.0 = MELDW, bit 6 (no motor overtemperature warning)
  - DB31, ... DBX94.1 = MELDW, bit 7 (no thermal overload in power unit warning)

- Case 2: Warning of warning class B (only in interface mode "SIMODRIVE 611U", p2038 = 1)
  - DB31, ... DBX94.0 == 1 and DBX94.1 == 1, if the following applies:
    - Cyclic drive telegram, status word 1 (ZSW1), bits 11/12 == 2 (warning class B)

The interface signals are derived from the warning of warning class B if there is no specific information from the message word.

An alarm is displayed. Alarm number = 200.000 + alarm value (r2124)

For a detailed description of the motor temperature monitoring setting, see:

References:

- S120 Commissioning Manual, Section "Commissioning" > "Temperature sensors for SINAMICS components"
- S120 Function Manual, Section "Monitoring and protective functions"
- S120 List Manual
  - MELDW, bit 6 ≙ BO: r2135.14 → function diagram: 2548, 8016
  - MELDW, bit 7 ≙ BO: r2135.15 → function diagram: 2548, 2452, 2456, 8016

DB31, ... DBX94.2 (run-up completed)

The actual speed value is within the parameterized tolerance band again after changing the speed setpoint. The run-up procedure is now completed.

Any subsequent speed fluctuations, also outside the tolerance band, e.g. due to load changes, will not affect the interface signal.

DB31, ... DBX94.3 (|M_d| < M_{dx})

The absolute value of the current torque |M_d| is less than the parameterized threshold torque M_{dx} (torque threshold value 2, p2194).

The threshold torque is set as a percentage [%] of the current speed-dependent torque limitation.

DB31, ... DBX94.4 (|n_{act}| < n_{min})

The actual speed value n_{act} is less than n_{min} (speed threshold value 3, p2161).
DB31, DBX94.5 \((|n_{\text{act}}| < n_x)\)

The actual speed value \(n_{\text{act}}\) is less than \(n_x\) (speed threshold value 2, p2155).

DB31, DBX94.6 \((n_{\text{act}} = n_{\text{set}})\)

The actual speed value is within the tolerance band (p2163) surrounding the speed setpoint.

DB31, DBX95.7 \((\text{warning of warning class C is pending})\)

The drive signals that a warning of warning class C is pending.

2.3 Functions

2.3.1 Screen refresh behavior for overload - only 840D sl

There are part programs, where the main run must wait until the preprocessing run provides new blocks.

Preprocessing run and display refresh compete for NC processor time.

The following machine data can be used to set how the NC should behave if the preprocessing run is too slow:

MD10131 $MN_SUPPRESS_SCREEN_REFRESH

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>If the preprocessing run of a channel is too slow, the display is not refreshed in all channels.</td>
</tr>
<tr>
<td>1</td>
<td>If the preprocessing run of a channel is too slow, the display is only refreshed in the time critical channels in order to obtain processor time for the preprocessing run.</td>
</tr>
<tr>
<td>2</td>
<td>Display refresh is not suppressed.</td>
</tr>
</tbody>
</table>
2.3.2 Settings for involute interpolation - only 840D sl

Introduction

The involute of the circle is a curve traced out from the end point on a "piece of string" unwinding from the curve. Involute interpolation allows trajectories along an involute.

![Figure 2-6 Involute (unwound from base circle)](image)

Programming

A general description of how to program involute interpolation can be found in:

References:
Programming Manual, Fundamentals

In addition to the programmed parameters, machine data is relevant in two instances of involute interpolation; this data may need to be set by the machine manufacturer / end user.
Accuracy

If the programmed end point does not lie exactly on the involute defined by the starting point, interpolation takes place between the two involutes defined by the starting and end points (see illustration below).

The maximum deviation of the end point is determined by the machine data:

MD21015 $MC_INVOLUTE_RADIUS_DELTA (end point monitoring for involute)

Figure 2-7 MD21015 specifies the max. permissible deviation
Limit angle

If AR is used to program an involute leading to the base circle with an angle of rotation that is greater than the maximum possible value, an alarm is output and program execution aborted.

![Diagram of involute and angle](image)

Figure 2-8 Limited angle of rotation towards base circle

The alarm display can be suppressed using the following parameter settings:

MD21016 $MC_INVOLUTE_AUTO_ANGLE_LIMIT = TRUE (automatic angle limitation for involute interpolation)

The programmed angle of rotation is then also limited automatically and the interpolated path ends at the point at which the involute meets the base circle. This, for example, makes it easier to program an involute which starts at a point outside the base circle and ends directly on it.

Tool radius compensation

2 1/2 D tool radius compensation is the only tool radius compensation function permitted for involutes. If 3D tool radius compensation is active (both circumferential and face milling), when an involute is programmed, machining is interrupted with alarm 10782.

With 2 1/2 D tool radius compensation, the plane of the involute must lie in the compensation plane. or else alarm 10781 will be generated. It is however permissible to program an additional helical component for an involute in the compensation plane.

Dynamic response

Involutes that begin or end on the base circle have an infinite curvature at this point. To ensure that the velocity is adequately limited at this point when tool radius compensation is active, without reducing it too far at other points, the "Velocity limitation profile" function must be activated:
MD28530 $MC_MM_PATH_VELO_SEGMENTS $1 (number of memory elements for limiting the path velocity)

A setting of 5 is recommended. This setting need not be made if only involute sections are used which have radii of curvature that change over a relatively small area.

2.3.3 Activate DEFAULT memory - only 840D sl

GUD start values

The DEF... / REDEF... NC commands can be used to assign default settings to global user data (GUD). To make these default values available at the parameterized initialization time, e.g. with the attribute INIPO, after power on, they must be saved permanently in the system. The required memory space must be enabled using the following machine data:

MD18150 $MM_GUD_VALUES_MEM (non-volatile memory space for GUD values)

References:

Function Manual, Extended Functions; S7: "Memory Configuration"
Programming Manual, Job Planning

2.3.4 Read and write PLC variable - only 840D sl

High-speed data channel

For high-speed exchange of information between the PLC and NC, a memory area is reserved in the communications buffer on these modules (dual-port RAM). Variables of any type (I/O, DB, DW, flags) may be exchanged within this memory area.

The PLC accesses this memory using 'Function Calls' (FC) while the NC uses system variables.

Organization of memory area

The user's programming engineer (NC and PLC) is responsible for organizing (structuring) this memory area.

Every storage position in the memory can be addressed provided that the limit is selected according to the appropriate data format (i.e. a DWORD for a 4-byte limit, a WORD for a 2-byte limit, etc.).

The memory is accessed via the data type and the position offset within the memory area.
Access from NC

System variables are available in the NC for fast access to PLC variables from a part program or synchronized action. The data is read/written directly by the NC. The data type results from the identifier of the system variables. The position within the memory area is specified as index in bytes.

<table>
<thead>
<tr>
<th>System variable</th>
<th>Data type</th>
<th>Range of values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_DBB[index]</td>
<td>Byte (8 bits)</td>
<td>0 ≤ x ≤ 255</td>
</tr>
<tr>
<td>$A_DBW[index]</td>
<td>Word (16 bits)</td>
<td>-32768 ≤ x ≤ 32767</td>
</tr>
<tr>
<td>$A_DBD[index]</td>
<td>Double word (32 bits)</td>
<td>-2147483648 ≤ x ≤ 2147483647</td>
</tr>
<tr>
<td>$A_DBR[index]</td>
<td>Floating point (32 bits)</td>
<td>±(1.5·10^-45 ≤ x ≤ 3.4·10^38)</td>
</tr>
</tbody>
</table>

Access from PLC

The PLC uses function calls (FC) to access the memory. The data is read and written immediately in the DPR with the FC and not just at the beginning of the PLC cycle. Data type and position in the memory area are transferred as parameters to the FC.

Supplementary conditions

- The structuring of the DPR memory area is the sole responsibility of the user. No checks are made for matching configuration.
- A total of 4096 bytes are available in the input and output directions.
- Single-bit operations are not supported and must be linked back to byte operations by the user.
- Since the contents of variables are manipulated directly in the communications buffer, the user must remember that intermediate changes in values occur as a result of multiple access operations where a variable is evaluated several times or when variables are linked (i.e. it may be necessary to store values temporarily in local variables or R parameters or to set up a semaphore).
• The user's programming engineer is responsible for coordinating access operations to the communications buffer from different channels.

• Data consistency can be guaranteed only for access operations up to 16 bits (byte and word). The user is solely responsible for ensuring consistent transmission of 32-bit variables (double and real). A simple semaphore mechanism is available in the PLC for this purpose.

• The PLC stores data in 'Little Endian' format in the DPR.

• Values transferred with $A\_DBR$ are subject to data conversion and hence to loss of accuracy. The data format for floating-point numbers is DOUBLE (64 bits) in the NC, but only FLOAT (32 bits) in the PLC. The format used for storage in the dual-port RAM is FLOAT. Conversion takes place respectively before/after storage in the dual-port RAM. If a read/write access is made from the NC to a variable in the dual-port RAM, the conversion is performed twice. It is impossible to prevent differences between read and written values because the data is stored in both formats.

**Example**

Bypassing the problem by means of comparison on "EPSILON" (minor deviation)

```
DEF REAL DBR
DEF REAL EPSILON = 0.00001
$A\_DBR[0]=145.145
G4 F2
STOPRE
DBR=$A\_DBR[0]
IF ( ABS(DBR/145.145-1.0) < EPSILON ) GOTOF ENDE
MSG ( "error" )
M0
END:
M30
```

**Activation**

The maximum number of simultaneously writable output variables is adjustable via: MD28150 $MC\_MM\_NUM\_VDIVAR\_ELEMENTS$ (number of elements for writing PLC variables)

**Example**

A variable of type WORD is to be transferred from the PLC to the NC.

The position offset within the NC input (PLC output area) should be the fourth byte. The position offset must be a whole-number multiple of the data width.

**Writing from PLC:**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>. . . CALL FC21 ( Enable :=M10.0, ; if TRUE, then FC21 active</td>
<td></td>
</tr>
</tbody>
</table>
Behavior during POWER ON, block search

The DPR communications buffer is initialized during "POWER ON". During a "block search", the PLC variable outputs are collected and transferred to the DPR communications buffer with the approach block (analogous to writing of analog and digital outputs). Other status transitions have no effect in this respect.

References

A detailed description of the data exchange by the PLC with FC 21 can be found in: SINUMERIK 840D sl: Section "FC21: transfer PLC NCK data exchange (Page 1037)"

2.3.5 Access protection via password and keyswitch

Access authorization

Access to functions, programs and data is user-oriented and controlled via 8 hierarchical protection levels. These are subdivided into:

- Password levels for Siemens, machine manufacturer and end user
- Keyswitch positions for end user
Multi-level security concept
A multi-level security concept to regulate access rights is available in the form of password levels and keyswitch settings.

<table>
<thead>
<tr>
<th>Protection level</th>
<th>Type</th>
<th>User</th>
<th>Access to (examples)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Password</td>
<td>Siemens</td>
<td>All functions, programs and data</td>
</tr>
<tr>
<td>1</td>
<td>Password</td>
<td>Machine manufacturer: Development</td>
<td>defined functions, programs and data; for example: entering options</td>
</tr>
<tr>
<td>2</td>
<td>Password</td>
<td>Machine manufacturer: Start-up engineer</td>
<td>defined functions, programs and data; for example: Bulk of machine data</td>
</tr>
<tr>
<td>3</td>
<td>Password</td>
<td>End user: Service</td>
<td>Assigned functions, programs and data</td>
</tr>
<tr>
<td>4</td>
<td>Keyswitch position 3</td>
<td>End user: Programmer, machine setter</td>
<td>less than the protection level 0 to 3; established by the machine manufacturer or end user</td>
</tr>
<tr>
<td>5</td>
<td>Keyswitch position 2</td>
<td>End user: Skilled operator without programming knowledge</td>
<td>less than the protection level 0 to 3; established by the end user</td>
</tr>
<tr>
<td>6</td>
<td>Keyswitch position 1</td>
<td>End user: Trained operator without programming knowledge</td>
<td>Example: Program selection only, tool wear entry, and work offset entry</td>
</tr>
<tr>
<td>7</td>
<td>Keyswitch position 0</td>
<td>End user: Semi-skilled operator</td>
<td>Example: no inputs and program selection possible, only machine control panel operable</td>
</tr>
</tbody>
</table>

Access features

- Protection level 0 provides the greatest number of access rights, protection level 7 the least.
- If certain access rights are granted to a protection level, these protection rights automatically apply to any higher protection levels.
- Conversely, protection rights for a certain protection level can only be altered from a higher protection level.
- Access rights for protection levels 0 to 3 are permanently assigned by Siemens and cannot be altered (default).
- Access rights can be set by querying the current keyswitch positions and comparing the passwords entered. When a password is entered it overwrites the access rights of the keyswitch position.
Options can be protected on each protection level. However, option data can only be entered in protection levels 0 and 1.

Access rights for protection levels 4 to 7 are only suggestions and can be altered by the machine tool manufacturer or end user.

2.3.5.1 Password

Set password

The password for a protection level (0 – 3) is entered via the HMI user interface.

Example:
DIAGNOSTIC operating area, softkey: SET PASSWORD

References:
Commissioning Manual SINUMERIK 840D sl base software and HMI sl

Delete password

Access rights assigned by means of setting a password remain effective until they are explicitly revoked by deleting the password.

Example:
DIAGNOSTIC operating area, softkey: DELETE PASSWORD

References:
Commissioning Manual SINUMERIK 840D sl base software and HMI sl

Note
Access rights and password status (set/deleted) are not affected by POWER OFF/ON!

Maximum number of characters

A password may contain up to eight characters. We recommend that you confine yourself to the characters available on the operator panel front when defining the password. Where a password consists of less than eight characters, the additional characters are interpreted as blanks.
Defaults

The following default passwords are defined for protection levels 1 to 3:

- Protection level 1: SUNRISE
- Protection level 2: EVENING
- Protection level 3: CUSTOMER

Note

Following NC-CPU ramp-up in commissioning mode (NCK commissioning switch: position 1) the passwords for protection levels 1 – 3 are reset to the default settings. For reasons of data protection, we strongly recommend that you change the default settings.

2.3.5.2 Keyswitch positions (DB10, DBX56.4 to 7)

Key switch

The keyswitch has four positions, to which protection levels 4 to 7 are assigned. The keyswitch comprises a number of keys in a variety of colors which can be set to different switch positions.

<table>
<thead>
<tr>
<th>Switch position</th>
<th>Retraction pos.</th>
<th>DB10, DBX56</th>
<th>Protection level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position 0</td>
<td>-</td>
<td>Bit 4</td>
<td>7</td>
</tr>
<tr>
<td>Position 1</td>
<td>0 or 1 black key</td>
<td>Bit 5</td>
<td>6</td>
</tr>
<tr>
<td>Position 2</td>
<td>0 or 1 or 2 green key</td>
<td>Bit 6</td>
<td>5</td>
</tr>
<tr>
<td>Position 3</td>
<td>0 or 1 or 2 or 3 red key</td>
<td>Bit 7</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 2-10 Switch positions 0 to 3

Switch positions

Switch position 0 has the most restricted access rights. Switch position 3 has the least restricted access rights.

DB10, DBX56.4 / .5 / .6 / .7 (switch positions 0 / 1 / 2 / 3)

Machine-specific enables for access to programs, data and functions can be assigned to the switch positions. For detailed information, please refer to:
2.3 Functions

Default settings via the PLC user program

The keyswitch positions are transferred to the NC/PLC interface via the basic PLC program. The corresponding interface signals can be modified via the PLC user program. In this context, from the point of view of the NC, only one switch position should ever be active, i.e. the corresponding interface signal set to 1. If, from the point of view of the NC, a number of switch positions are active at the same time, switch position 3, i.e. the keyswitch position with the least restricted access rights, will be activated internally by the NC.

2.3.5.3 Parameterizable protection levels

Parameterizable protection level

The parameter level can be freely parameterized for a variety of functions and data areas. The protection level is set via operator-panel machine data, designated as follows:

$MM_USER_CLASS_ <Function_DataArea>

Examples:

$MM_USER_CLASS_READ_TOA Read tool offsets
$MM_USER_CLASS_WRITE_TOA Write tool offsets
$MM_USER_CLASS_READ_PROGRAM Read part programs
$MM_USER_CLASS_WRITE_PROGRAM Write/edit part programs

Default values

On delivery or following standard commissioning, with very few exceptions, the default value for the protection level will be set to 7, i.e. the lowest protection level.
2.3.6 Switching over motor/drive data sets

2.3.6.1 General Information

Motor and drive data sets

For optimum adaptation to the particular machining situation or because of different machine configurations, it may be necessary that several different data sets are available in a drive for motors, drive parameters and encoders. The creation of the basic data sets of the drive objects is performed during startup with the aid of the "Drive wizard".

Note

References

Commissioning Manual: CNC: NCK, PLC, Drive, Section "Commissioning NC-controlled drives"

The following duplication and management of the data sets is performed via the user interface:

SINUMERIK Operate: Operating area "Start-up" > "Drive system" > "Drives" > "Data sets"

The activation of the motor data set (MDS) or drive data set (DDS) required for a machine axis in a specific machining situation, must be made from the PLC user program via the interfaces described below.

Axial NC/PLC interface

The interfaces in the axial NC/PLC interface for switching the motor and drive data sets is divided into three areas:

- Formatting interface (Page 65)
- Request interface (Page 66)
- Display interface (Page 67)

2.3.6.2 Formatting interfaces

Formatting

The formatting interface is used to set which bits of the request and display interface are used to address the motor data sets (MDS) and which are used to address the drive data sets (DDS):

DB31, ... DBX130.0 - 4, with bit x = <value>

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Bit position for motor data set (MDS) – or invalid bit position</td>
</tr>
<tr>
<td>1</td>
<td>Bit position for drive data set (DDS)</td>
</tr>
</tbody>
</table>
Motor and drive data sets in the drive

Formatting depends on the number of motor data sets (MDS) and drive data sets (DDS) in the drive. The number can be determined using the following drive parameters:

- p0130 (number of motor data sets)
- p0180 (number drive data sets)

Validity of the interfaces

After powering up, as soon as the control has received all the required information from the drive and this has been evaluated by the NC, the request (Page 66) and display interfaces (Page 67) are shown as valid:

DB31, ... DBX130.7 == 1 (request and display interfaces valid)

If no or incompatible information is transferred from the drive, the request and display interfaces are shown as invalid.

Note

With invalid request and display interfaces, it however remains the sole responsibility of the user / machine manufacturer to perform a data set switchover via the invalid request and display interfaces.

See also

- Example (Page 67)
- Overview of the interfaces (Page 68)

2.3.6.3 Request interface

The switchover to a new motor (MDS) and/or drive data sets (DDS) is requested via the interface:

DB31, ... DBX21.0 - .4 = <MDS/DDS index>

Value range

Addressing a motor or drive data set n, with n = 1, 2, 3, ..., is realized based on its index i, with i = n - 1 = 0, 1, 2, ...

- Motor data sets: MDS[ 0, 1, 2, ... 15 ]
- Drive data sets: DDS[ 0, 1, 2, ... 31 ]

Formatting interfaces

Formatting the request interface, i.e. which bits are used to address the motor data sets (MDS) – and which are used to address the drive data sets (DDS) is set via the formatting interface (Page 65).
Motor and drive data sets in the drive
The number of motor data sets (MDS) and drive data sets (DDS) in the drive can be determined using the following drive parameters:

- p0130 (number of motor data sets)
- p0180 (number drive data sets)

Motor data set (MDS) for main spindle drives
For main spindle drives, the following association applies:

- MDS[ 0 ] → star operation
- MDS[ 1 ] → delta operation

2.3.6.4 Display interface
The active motor data set (MDS) and drive data set (DDS) are displayed via the interface:

DB31, ... DBX93.0 - .4 == <MDS / DDS index>

Value range and formatting are identical to the request interface (Page 66).

2.3.6.5 Example
Two motor data sets (MDS) and two drive data sets (DDS) are available in the drive for each motor data set. This corresponds to "No.": 9 of the possible data set combinations displayed in the Figure 2-11 Principle of the motor/drive data set switchover (Page 69).

Format
Bit positions for drive data set switchover (DDS):

- DB31, ... DBX130.0 == 1

Bit positions for motor data set switchover (MDS):

- DB31, ... DBX130.1 == 0

Invalid bit positions:

- DB31, ... DBX130.2 == 0
- DB31, ... DBX130.3 == 0
- DB31, ... DBX130.4 == 0

Interfaces of the drive data sets (DDS)
Relevant bit positions of the request and display interfaces:

- DB31, ... DBX21.0 / DBX93.0
  - DB31, ... DBX21.0 / DBX93.0 == 0 ⇒ 1st drive data set DDS[0]
  - DB31, ... DBX21.0 / DBX93.0 == 1 ⇒ 2nd drive data set DDS[1]
Interfaces of the motor data sets (MDS)
Relevant bit positions of the request and display interfaces:

- DB31, ... DBX21.1 / DBX93.1
  - DB31, ... DBX21.1 / DBX93.1 == 0 ⇒ 1st motor data set MDS[0]
  - DB31, ... DBX21.1 / DBX93.1 == 1 ⇒ 2nd motor data set MDS[1]

Invalid bit positions (MDS/DDS)
Invalid bit positions of the request and display interfaces:

- DB31, ... DBX21.1 / DBX93.2 == 0
- DB31, ... DBX21.1 / DBX93.3 == 0
- DB31, ... DBX21.1 / DBX93.4 == 0

See also
Overview of the interfaces (Page 68)

2.3.6.6 Overview of the interfaces

Table 2-1 Configurable MDS / DDS combinations

<table>
<thead>
<tr>
<th>Number of MDS (motors)</th>
<th>Number of DDS (drives) per MDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 ... 32</td>
</tr>
<tr>
<td>2</td>
<td>1, 2, 4, 8, 16</td>
</tr>
<tr>
<td>3</td>
<td>1, 2, 4, 8</td>
</tr>
<tr>
<td>4</td>
<td>1, 2, 4, 8</td>
</tr>
<tr>
<td>5</td>
<td>1, 2, 4</td>
</tr>
<tr>
<td>6</td>
<td>1, 2, 4</td>
</tr>
<tr>
<td>7</td>
<td>1, 2, 4</td>
</tr>
<tr>
<td>8</td>
<td>1, 2, 4</td>
</tr>
<tr>
<td>9</td>
<td>1, 2</td>
</tr>
<tr>
<td>10</td>
<td>1, 2</td>
</tr>
<tr>
<td>11</td>
<td>1, 2</td>
</tr>
<tr>
<td>12</td>
<td>1, 2</td>
</tr>
<tr>
<td>13</td>
<td>1, 2</td>
</tr>
<tr>
<td>14</td>
<td>1, 2</td>
</tr>
<tr>
<td>15</td>
<td>1, 2</td>
</tr>
<tr>
<td>16</td>
<td>1, 2</td>
</tr>
</tbody>
</table>
Supplementary conditions

Variable number of drive data sets for the "last" motor data set

The "last" motor data set is the motor data set with the highest number or index.

Generally, the same number of drive data sets is created for each motor data set (number of "DDS per MDS") in the drive. Only the "last" motor data set can differ from this; any number (a) of drive data sets can be parameterized:

\[ 1 \leq a \leq (\text{number of "DDS per MDS"}) \]

Example

4 motor data sets (MDS) and 8 drive data sets (DDS) per motor data set (DDS per MDS) are to be parameterized. This corresponds to "No.": 22 of the possible data set combination displayed in the Figure 2-11 Principle of the motor/drive data set switchover (Page 69):

- Motor data sets: MDS[0], MDS[1], ... MDS[3] ("last" motor data set)
- Drive data sets per motor data set: DDS[0] ... DDS[7]
The number of drive data sets for the individual motor data sets is therefore:

<table>
<thead>
<tr>
<th>Motor data set (MDS)</th>
<th>Number of drive data sets (DDS) per motor data set (MDS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDS[ 0 ] ... MDS[ 2 ]</td>
<td>8</td>
</tr>
<tr>
<td>MDS[ 3 ]</td>
<td>1 - 8</td>
</tr>
</tbody>
</table>

Switchover instant: Drive parameter set

In principle it is possible to switch over drive parameter sets at any point in time. While an axis is traversing, especially when switching over speed controller parameters and the motor speed scaling, torque jumps/steps can occur. We therefore recommend that a drive parameter set is only switched over for stationary states, especially when the axis is at a standstill.

See also

Overview of the interfaces (Page 68)

2.4 Examples

2.4.1 Parameter set changeover

Parameter set changeover

A parameter set changeover is performed to change the position control gain \( (K_v \text{factor}) \) from \( K_v = 4.0 \) to \( K_v = 0.5 \) for machine axis X1.

Preconditions

The parameter set changeover must be enabled by the machine data:

\[
\text{MD35590 $MA\_PARAMSET\_CHANGE\_ENABLE \[AX1\] = 1 or 2 (parameter set change possible)}
\]

The 1st parameter set for machine axis X1 is set, in accordance with machine data with index "0" NC/PLC interface:

\[
\text{DB31, … DBX9.0 - DBX9.2 = 0 (controller parameter set)}
\]

Parameter-set-dependent machine data

Parameter-set-dependent machine data are set as follows:

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD32200 $MA_POSCTRL_GAIN [0, AX1] = 4.0</td>
<td>( K_v ) setting for parameter set 1</td>
</tr>
<tr>
<td>MD32200 $MA_POSCTRL_GAIN [1, AX1] = 2.0</td>
<td>( K_v ) setting for parameter set 2</td>
</tr>
</tbody>
</table>
### Switchover

In order to switch over the position-control gain, the PLC user program selects the 4th parameter set for machine axis AX1.

- **Request by PLC user program:**
  
  DB31, ... DBX9.0 – DBX9.2 = 3 (parameter set servo)
  
  - A request to change over to the 4th parameter set is sent for machine axis AX1.
  - The parameter set is changed over once a delay has elapsed.
  - Parameter set 4 is now active, in accordance with machine data with index "3"

- **Feedback by NC:**
  
  DB31, ... DBX69.0 – DBX69.2 = 3 (parameter set servo)
  
  - The NC confirms/acknowledges the parameter-set changeover.
2.5 Data lists

2.5.1 Machine data

2.5.1.1 Display machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MM_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9000</td>
<td>LCD_CONTRAST</td>
<td>SINUMERIK Operate Contrast</td>
</tr>
<tr>
<td>9001</td>
<td>DISPLAY_TYPE</td>
<td>Monitor type</td>
</tr>
<tr>
<td>9004</td>
<td>DISPLAY_RESOLUTION</td>
<td>Display resolution</td>
</tr>
</tbody>
</table>

2.5.1.2 NC-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10350</td>
<td>FASTIO_DIG_NUM_INPUTS</td>
<td>Number of active digital NCK input bytes</td>
</tr>
<tr>
<td>10360</td>
<td>FASTIO_DIG_NUM_OUTPUTS</td>
<td>Number of active digital NCK output bytes</td>
</tr>
<tr>
<td>10361</td>
<td>FASTIO_DIG_SHORT_CIRCUIT</td>
<td>Short-circuit digital inputs and outputs</td>
</tr>
<tr>
<td>11120</td>
<td>LUD_EXTENDED_SCOPE</td>
<td>Activate programglobal variables (PUD)</td>
</tr>
<tr>
<td>11270</td>
<td>DEFAULT_VALUES_MEM_MSK</td>
<td>Activ. Function: Save DEFAULT values of GUD.</td>
</tr>
<tr>
<td>18150</td>
<td>MM_GUD_VALUES_MEM</td>
<td>Reserve memory space for GUD</td>
</tr>
</tbody>
</table>

2.5.1.3 Channelspecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>21015</td>
<td>INVOLUTE_RADIUS_DELTA</td>
<td>NC start disable without reference point</td>
</tr>
<tr>
<td>21016</td>
<td>INVOLUTE_AUTO_ANGLE_LIMIT</td>
<td>Automatic angle limitation for involute interpolation</td>
</tr>
<tr>
<td>27800</td>
<td>TECHNOLOGY_MODE</td>
<td>Technology in channel</td>
</tr>
<tr>
<td>28150</td>
<td>MM_NUM_VDIVAR_ELEMENTS</td>
<td>Number of write elements for PLC variables</td>
</tr>
<tr>
<td>28530</td>
<td>MM_PATH_VELO_SEGMENTS</td>
<td>Number of storage elements for limiting path velocity in block</td>
</tr>
</tbody>
</table>

2.5.1.4 Axis/spindlespecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>30350</td>
<td>SIMU_AX_VDI_OUTPUT</td>
<td>Output of axis signals for simulation axes</td>
</tr>
<tr>
<td>33050</td>
<td>LUBRICATION_DIST</td>
<td>Lubrication pulse distance</td>
</tr>
</tbody>
</table>
### 2.5 Data lists

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA___</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>35590</td>
<td>PARAMSET_CHANGE_ENABLE</td>
<td>Parameter set definition possible from PLC</td>
</tr>
<tr>
<td>36060</td>
<td>STANDSTILL_VELO_TOL</td>
<td>Maximum velocity/speed when axis/spindle stationary</td>
</tr>
<tr>
<td>36610</td>
<td>AX_EMERGENCY_STOP_TIME</td>
<td>Length of the braking ramp for error states</td>
</tr>
<tr>
<td>36620</td>
<td>SERVO_DISABLE_DELAY_TIME</td>
<td>Cutout delay servo enable</td>
</tr>
</tbody>
</table>
2.5.2 System variables

<table>
<thead>
<tr>
<th>Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_FUMB</td>
<td>Unassigned part program memory (Free User Memory Buffer)</td>
</tr>
<tr>
<td>$A_DBB[n]</td>
<td>Data on PLC (data type BYTE)</td>
</tr>
<tr>
<td>$A_DBW[n]</td>
<td>Data on PLC (WORD type data)</td>
</tr>
<tr>
<td>$A_DBD[n]</td>
<td>Data on PLC (DWORD type data)</td>
</tr>
<tr>
<td>$A_DBR[n]</td>
<td>Data on PLC (REAL type data)</td>
</tr>
</tbody>
</table>

2.5.3 Signals

2.5.3.1 Signals to NC

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyswitch setting 0 to 3</td>
<td>DB10.DBX56.4 - 7</td>
<td>DB2600.DBX0.4 - 7</td>
</tr>
</tbody>
</table>

2.5.3.2 Signals from NC

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote diagnostics active (HMI alarm is pending)</td>
<td>DB10.DBX103.0</td>
<td></td>
</tr>
<tr>
<td>AT box ready</td>
<td>DB10.DBX103.5</td>
<td></td>
</tr>
<tr>
<td>HMI temperature limit</td>
<td>DB10.DBX103.6</td>
<td></td>
</tr>
<tr>
<td>HMI battery alarm</td>
<td>DB10.DBX103.7</td>
<td></td>
</tr>
<tr>
<td>NC ready</td>
<td>DB10.DBX104.7</td>
<td></td>
</tr>
<tr>
<td>Operator panel 2: &quot;ready&quot;</td>
<td>DB10.DBX108.1</td>
<td></td>
</tr>
<tr>
<td>Operator panel at MPI: &quot;ready&quot;</td>
<td>DB10.DBX108.2</td>
<td></td>
</tr>
<tr>
<td>Operator panel at OPI: &quot;ready&quot;</td>
<td>DB10.DBX108.3</td>
<td>DB2700.DBX2.3</td>
</tr>
<tr>
<td>Drives in cyclic operation</td>
<td>DB10.DBX108.5</td>
<td>DB2700.DBX2.5</td>
</tr>
<tr>
<td>Drives ready</td>
<td>DB10.DBX108.6</td>
<td>DB2700.DBX2.6</td>
</tr>
<tr>
<td>NC Ready</td>
<td>DB10.DBX108.7</td>
<td>DB2700.DBX2.7</td>
</tr>
<tr>
<td>NC alarm is active</td>
<td>DB10.DBX109.0</td>
<td>DB2700.DBX3.0</td>
</tr>
<tr>
<td>NCU heat sink temperature alarm</td>
<td>DB10.DBX109.5</td>
<td></td>
</tr>
<tr>
<td>Air temperature alarm</td>
<td>DB10.DBX109.6</td>
<td>DB2700.DBX3.6</td>
</tr>
<tr>
<td>NC battery alarm</td>
<td>DB10.DBX109.7</td>
<td></td>
</tr>
</tbody>
</table>
### 2.5.3.3 Signals to operator panel front

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen bright</td>
<td>DB19.DBX0.0</td>
<td>-</td>
</tr>
<tr>
<td>Screen dark</td>
<td>DB19.DBX0.1</td>
<td>DB1900.DBX5000.1</td>
</tr>
<tr>
<td>Key disable</td>
<td>DB19.DBX0.2</td>
<td>DB1900.DBX5000.2</td>
</tr>
<tr>
<td>Delete Cancel alarms (HMI Advanced only)</td>
<td>DB19.DBX0.3</td>
<td>-</td>
</tr>
<tr>
<td>Delete Recall alarms (HMI Advanced only)</td>
<td>DB19.DBX0.4</td>
<td>-</td>
</tr>
<tr>
<td>Actual value in the WCS, (1) / MCS (0)</td>
<td>DB19.DBX0.7</td>
<td>DB1900.DBX5000.7</td>
</tr>
<tr>
<td>Unload part program</td>
<td>DB19.DBX13.5</td>
<td>-</td>
</tr>
<tr>
<td>Load part program</td>
<td>DB19.DBX13.6</td>
<td>-</td>
</tr>
<tr>
<td>Part program selection</td>
<td>DB19.DBX13.7</td>
<td>DB1700.DBX1000.7</td>
</tr>
<tr>
<td>File system active (0) / passive (1)</td>
<td>DB19.DBX14.7</td>
<td>-</td>
</tr>
<tr>
<td>Part program handling: Number of the control data for user file names</td>
<td>DB19.DBB16</td>
<td>DB1700.DBB1001</td>
</tr>
<tr>
<td>Part program handling: Index of the file to be transferred from the user list</td>
<td>DB19.DBB17</td>
<td>DB1700.DBB1002</td>
</tr>
<tr>
<td>Mode change disable</td>
<td>DB19.DBX44.0</td>
<td>-</td>
</tr>
</tbody>
</table>

<sup>1)</sup> For SINUMERIK 828D, the bright/dark control is realized using DB1900.DBX5000.1:
- DB1900.DBX5000.1=0: Screen bright
- DB1900.DBX5000.1=1: Screen dark
- DB1900.DBX5000.0 has no significance for the screensaver function.

<sup>2)</sup> Bit 7: Always 1

### 2.5.3.4 Signals from operator panel front

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen is dark</td>
<td>DB19.DBX20.1</td>
<td>-</td>
</tr>
<tr>
<td>Switch over MCS/WCS</td>
<td>DB19.DBX20.7</td>
<td>DB1900.DBX0.7</td>
</tr>
<tr>
<td>Part program handling status: Error</td>
<td>DB19.DBX26.0</td>
<td>DB1700.DBX2000.2</td>
</tr>
<tr>
<td>Part program handling status: OK</td>
<td>DB19.DBX26.1</td>
<td>DB1700.DBX2000.1</td>
</tr>
<tr>
<td>Part program handling status: Active</td>
<td>DB19.DBX26.3</td>
<td>DB1700.DBX2000.3</td>
</tr>
<tr>
<td>Part program handling status: Unloading</td>
<td>DB19.DBX26.5</td>
<td>DB1700.DBX2000.5</td>
</tr>
<tr>
<td>Part program handling status: Loading</td>
<td>DB19.DBX26.6</td>
<td>DB1700.DBX2000.6</td>
</tr>
<tr>
<td>Part program handling status: Selection</td>
<td>DB19.DBX26.7</td>
<td>DB1700.DBX2000.7</td>
</tr>
<tr>
<td>FC9: Start &quot;Measuring in JOG&quot;</td>
<td>DB19.DBX42.0</td>
<td>-</td>
</tr>
<tr>
<td>FC9 Out: Active</td>
<td>DB19.DBX45.0</td>
<td>-</td>
</tr>
<tr>
<td>FC9 Out: Done</td>
<td>DB19.DBX45.1</td>
<td>-</td>
</tr>
<tr>
<td>FC9 Out: Error</td>
<td>DB19.DBX45.2</td>
<td>-</td>
</tr>
<tr>
<td>FC9 Out: StartErr</td>
<td>DB19.DBX45.3</td>
<td>-</td>
</tr>
</tbody>
</table>
### 2.5.3.5 Signals to channel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete distance to go (channelspecific)</td>
<td>DB21, ...DBX6.2</td>
<td>DB320x.DBX6.2</td>
</tr>
</tbody>
</table>

### 2.5.3.6 Signals from channel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel-specific NC alarm is present</td>
<td>DB21, ...DBX36.6</td>
<td>DB330x.DBX4.6</td>
</tr>
<tr>
<td>NC alarm with machining stop active</td>
<td>DB21, ...DBX36.7</td>
<td>DB330x.DBX4.7</td>
</tr>
<tr>
<td>Overstore active</td>
<td>DB21, ...DBX318.7</td>
<td></td>
</tr>
</tbody>
</table>

### 2.5.3.7 Signals to axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis/spindle disable</td>
<td>DB31, ...DBX1.3</td>
<td>DB380x.DBX1.3</td>
</tr>
<tr>
<td>Follow-up mode</td>
<td>DB31, ...DBX1.4</td>
<td>DB380x.DBX1.4</td>
</tr>
<tr>
<td>Position measuring system 1</td>
<td>DB31, ...DBX1.5</td>
<td>DB380x.DBX1.5</td>
</tr>
<tr>
<td>Position measuring system 2</td>
<td>DB31, ...DBX1.6</td>
<td>DB380x.DBX1.6</td>
</tr>
<tr>
<td>Controller enable</td>
<td>DB31, ...DBX2.1</td>
<td>DB380x.DBX2.1</td>
</tr>
<tr>
<td>Delete distance-to-go (axis-specific) / spindle reset</td>
<td>DB31, ...DBX2.2</td>
<td>DB380x.DBX2.2</td>
</tr>
<tr>
<td>Motor/drive data set: Request interface</td>
<td>DB31, ...DBX21.0 - 4</td>
<td>DB380x.DBX4001.0 - 4</td>
</tr>
<tr>
<td>Motor being selected</td>
<td>DB31, ...DBX21.5</td>
<td>DB380x.DBX4001.5</td>
</tr>
<tr>
<td>Speed controller integrator disable</td>
<td>DB31, ...DBX21.6</td>
<td>DB380x.DBX4001.6</td>
</tr>
<tr>
<td>Pulse enable</td>
<td>DB31, ...DBX21.7</td>
<td>DB380x.DBX4001.7</td>
</tr>
</tbody>
</table>

### 2.5.3.8 Signals from axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referenced/synchronized, encoder 1/2</td>
<td>DB31, ...DBX60.4/5</td>
<td>DB390x.DBX0.4/5</td>
</tr>
<tr>
<td>Traversing command minus/plus</td>
<td>DB31, ...DBX64.6/7</td>
<td>DB390x.DBX4.6/7</td>
</tr>
<tr>
<td>Follow up active</td>
<td>DB31, ...DBX61.3</td>
<td>DB390x.DBX1.3</td>
</tr>
<tr>
<td>Axis/spindle stationary (n &lt; n_{max})</td>
<td>DB31, ...DBX61.4</td>
<td>DB390x.DBX1.4</td>
</tr>
<tr>
<td>Position controller active</td>
<td>DB31, ...DBX61.5</td>
<td>DB390x.DBX1.5</td>
</tr>
<tr>
<td>Speed controller active</td>
<td>DB31, ...DBX61.6</td>
<td>DB390x.DBX1.6</td>
</tr>
<tr>
<td>Current controller active</td>
<td>DB31, ...DBX61.7</td>
<td>DB390x.DBX1.7</td>
</tr>
<tr>
<td>Lubrication pulse</td>
<td>DB31, ...DBX76.0</td>
<td>DB390x.DBX1002.0</td>
</tr>
<tr>
<td>Motor/drive data set: Display interface</td>
<td>DB31, ...DBX93.0 - 4</td>
<td>DB390x.DBX4001.0 - 4</td>
</tr>
<tr>
<td>Drive ready</td>
<td>DB31, ...DBX93.5</td>
<td>DB390x.DBX4001.5</td>
</tr>
<tr>
<td>Speed controller integrator disabled</td>
<td>DB31, ...DBX93.6</td>
<td>DB390x.DBX4001.6</td>
</tr>
</tbody>
</table>
## 2.5 Data lists

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulses enabled</td>
<td>DB31, ..., DBX93.7</td>
<td>DB390x.DBX4001.7</td>
</tr>
<tr>
<td>Motor temperature prewarning</td>
<td>DB31, ..., DBX94.0</td>
<td>DB390x.DBX4002.0</td>
</tr>
<tr>
<td>Heat sink temperature prewarning</td>
<td>DB31, ..., DBX94.1</td>
<td>DB390x.DBX4002.1</td>
</tr>
<tr>
<td>Run-up completed</td>
<td>DB31, ..., DBX94.2</td>
<td>DB390x.DBX4002.2</td>
</tr>
<tr>
<td></td>
<td>M_d</td>
<td>&lt; M_{in}</td>
</tr>
<tr>
<td></td>
<td>n_{act}</td>
<td>&lt; n_{min}</td>
</tr>
<tr>
<td></td>
<td>n_{act}</td>
<td>&lt; n_{x}</td>
</tr>
<tr>
<td>n_{act} = n_{set}</td>
<td>DB31, ..., DBX94.6</td>
<td>DB390x.DBX4002.6</td>
</tr>
<tr>
<td>Motor/drive data set: Formatting interface</td>
<td>DB31, ..., DBX130.0 - 4</td>
<td>DB390x.DBX4008.0 - 4</td>
</tr>
<tr>
<td>Motor/drive data set: Formatting is valid</td>
<td>DB31, ..., DBX130.7</td>
<td>DB390x.DBX4008.7</td>
</tr>
</tbody>
</table>
3.1 Brief description

3.1.1 Axis monitoring functions

Comprehensive monitoring functions are present in the controller for protection of people and machines:

- Contour monitoring
- Positioning monitoring
- Zero-speed monitoring
- Clamping monitoring
- Speed-setpoint monitoring
- Actual-velocity monitoring
- Measuring system monitoring
- Limit-switch monitoring
- Monitoring of the working area limitation

3.1.2 Protection zones

With the help of protection zones, elements of the machine (e.g. spindle chuck, tool changer, toolholder, tailstock, movable probe, etc.) and the workpiece can be protected against collisions.

During automatic execution of part programs in the AUTOMATIC or MDA mode, the NC checks at the start of every part program block whether a collision between protection zones can occur upon moving along the programmed path.

After manual deactivation of an active protection zone, traversing can be performed in this zone. After leaving the protection zone, the protection zone automatically becomes active again.

The definition, activation and deactivation of protection zones takes place via part program instructions.
3.2 Axis monitoring functions

3.2.1 Contour monitoring

3.2.1.1 Contour error

Contour errors are caused by signal distortions in the position control loop.

Signal distortions can be linear or non-linear.

Linear signal distortions

Linear signal distortions are caused by:

- Speed and position controller not being set optimally
- Different servo gain factors ($K_v$) of the feed axes involved in creating the path
  With the same servo gain factor ($K_v$) for two linear-interpolated axes, the actual position follows the set position along the same path but with a time delay. With different servo gain factors ($K_v$), a parallel offset arises between the set and actual path.
- Unequal dynamic response of the feed drives
  Unequal drive dynamic responses lead to path deviations especially on contour changes. Circles are distorted into ellipses by unequal dynamic responses of the two feed drives.

Non-linear signal distortions

Non-linear signal distortions are caused by:

- Activation of the current limitation within the machining area
- Activation of the limitation of the speed setpoint
- Backlash within and/or outside the position control loop
  When traversing a circular path, contour errors occur primarily due to the reversal error and friction.
  During motion along straight lines, a contour error arises due to a reversal error outside the position control loop, e.g. due to a tilting milling spindle. This causes a parallel offset between the actual and the set contour. The shallower the gradient of the straight line, the larger the offset.
- Nonlinear friction behavior of slide guides

3.2.1.2 Following-error monitoring

Function

In control engineering terms, traversing along a machine axis always produces a certain following error, i.e. a difference between the set and actual position.
The following error that arises depends on:

- Position control loop gain
  MD32200 $MA_POSCTRL_GAIN (servo gain factor)

- Maximum acceleration
  MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration)

- Maximum velocity
  MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

- With activated feedforward control:
  Precision of the path model and the parameters:
  MD32610 $MA_VELO_FFW_WEIGHT (factor for the velocity feedforward control)
  MD32800 $MA_EQUIV_CURRCTRL_TIME (equivalent time constant current control loop for feedforward control)
  MD32810 $MA_EQUIV_SPEEDCTRL_TIME (equivalent time constant speed control loop for feedforward control)

In the acceleration phase, the following error initially increases when traversing along a machine axis. After a time depending on the parameterization of the position control loop, the following error then remains constant in the ideal case. Due to external influences, more or less large fluctuations in the following error always arise during a machining process. To prevent these fluctuations in the following error from triggering an alarm, a tolerance range within which the following error may change must be defined for the following-error monitoring:

MD36400 $MA_CONTOUR_TOL (Contour monitoring tolerance range)
Effectiveness

The following-error monitoring only operates with active position control and the following axis types:

- Linear axes with and without feedforward control
- Rotary axes with and without feedforward control
- Position-controlled spindles

Fault

If the configured tolerance limit is exceeded, the following alarm appears:

25050 "Axis <Axis name> Contour monitoring"

The affected axis/spindle is stopped via the configured braking ramp in follow-up mode:

MD36610 $MA_AX_EMERGENCY_STOP_TIME
(maximum time for braking ramp when an error occurs)
3.2.2 Positioning, zero speed and clamping monitoring

3.2.2.1 Correlation between positioning, zero-speed and clamping monitoring

Overview

The following overview shows the correlation between the positioning, zero speed and clamping monitoring functions:

![Graph showing the correlation between positioning, zero speed, and clamping monitoring](image)

3.2.2.2 Positioning monitoring

Function

At the end of a positioning operation:

- Set velocity = 0 AND
- DB31, ... DBX64.6/7 (motion command minus/plus) = 0

checks the position monitoring to ensure that the following error of every participating machine axis is smaller than the exact-stop fine tolerance during the delay time.

MD36010 $MA_STOP_LIMIT_FINE (exact stop fine)
MD36020 $MA_POSITIONING_TIME (delay time exact stop fine)

After reaching "Exact stop fine", the position monitoring is deactivated.

**Note**

The smaller the exact stop fine tolerance is, the longer the positioning operation takes and the longer the time until block change.

### Rules for MD setting

<table>
<thead>
<tr>
<th>MD36010 $MA_STOP_LIMIT_FINE</th>
<th>MD36020 $MA_POSITIONING_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large</td>
<td>Can be selected relatively short</td>
</tr>
<tr>
<td>Small</td>
<td>Must be selected relatively long</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MD32200 $MA_POSCTRL_GAIN (servo gain factor)</th>
<th>MD36020 $MA_POSITIONING_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>Must be selected relatively long</td>
</tr>
<tr>
<td>Large</td>
<td>Can be selected relatively short</td>
</tr>
</tbody>
</table>

### Effectiveness

The position monitoring only operates with active position control and the following axis types:

- Linear axes
- Rotary axes
- Position-controlled spindles

### Fault

If the configured position-monitoring time is exceeded, the following alarm appears:

25080 "Axis <Axis name> Position monitoring"

The affected axis is stopped via the configured braking ramp in follow-up mode:

MD36610 $MA_AX_EMERGENCY_STOP_TIME
(maximum time for braking ramp when an error occurs)

### 3.2.2.3 Zero-speed monitoring

### Function

At the end of a positioning operation:

- Set velocity = 0 **AND**
- DB31, ... DBX64.6/7 (motion command minus/plus) = 0
checks the zero-speed monitoring to ensure that the following error of every participating machine axis is smaller than the standstill tolerance during the delay time.

MD36040 $MA_STANDSTILL_DELAY_TIME (zero-speed monitoring delay time)
MD36030 $MA_STANDSTILL_POS_TOL (standstill tolerance)

After reaching the required exact-stop state, the positioning operation is completed:

DB31, ... DBX60.6/7 (position reached with exact stop coarse/fine) = 1

The position-monitoring function is deactivated and is replaced by the zero-speed monitoring. Zero-speed monitoring monitors the adherence to the standstill tolerance. If no new travel request is received, the machine axis must not depart from the standstill tolerance.

**Effectiveness**

The zero-speed monitoring only operates with active position control and the following axis types:

- Linear axes
- Rotary axes
- Position-controlled spindles

**Fault**

If the delay time and/or the standstill tolerance is exceeded, the following alarm appears:

25040 "Axis <Axis name> Zero-speed monitoring"

The affected axis is stopped via the configured braking ramp in follow-up mode:

MD36610 $MA_AX_EMERGENCY_STOP_TIME
(maximum time for braking ramp when an error occurs)

**3.2.2.4 Parameter set-dependent exact stop and standstill tolerance**

For adaptation to different machining situations and/or axis dynamics, e.g.:

- Operating state A: High precision, long machining time
- Operating state B: Lower precision, shorter machining time
- Changing of the mass relationships after gear change

the positioning tolerances:

- MD36000 $MA_STOP_LIMIT_COARSE (exact stop coarse)
- MD36010 $MA_STOP_LIMIT_FINE (exact stop fine)
- MD36030 $MA_STANDSTILL_POS_TOL (standstill tolerance)

can be weighted with a common factor depending on the parameter set:

MD36012 $MA_STOP_LIMIT_FACTOR (exact stop coarse/fine and standstill factor)

Because the factor applies in common for all three position tolerances, the relationship between the values remains constant.
3.2.2.5 Clamping monitoring

Function
For machine axes that are mechanically clamped upon completion of a positioning operation, larger motions can result from the clamping process (> standstill tolerance). As a result, standstill monitoring is replaced by clamping monitoring during the clamping process. Clamping monitoring monitors the adherence to the configured clamping tolerance:
MD36050 $MA.Clamp.Pos.Tol (clamping tolerance)

Activation
The clamping monitoring is activated by the following interface signal:
DB31, ... DBX2.3 (clamping in progress) = 1

Note
The clamping monitoring is not active in "follow-up mode" (DB31, ... DBX1.4 = 1).

Error case
If the clamping tolerance is exceeded, alarm 26000 "clamping monitoring" is displayed: The affected axis is stopped via the configured braking ramp in follow-up mode:
MD36610 $MA.AX.EMERGENCY.Stop.TIME (maximum time for braking ramp when a fault occurs)

Automatic stop to release the clamping
If a clamped axis must be traversed again in continuous-path mode, the NC stops the path motion for Look Ahead at the start of the motion block of the clamped axis until the clamped axis can once again be traversed. If the clamping is released before stopping, the path motion is not stopped.

Parameterization:
MD36052 $MA.Stop.On.Clamping = 'H01' (special function for clamped axis)

Note
The NC detects whether an axis is clamped based on the "servo enable" state of the axis:
DB31, ... DBX2.1 = 0: No servo enable ⇒ axis is clamped
DB31, ... DBX2.1 = 1: Servo enable ⇒ axis is not clamped
Requirements for the PLC user program

- The axis is always removed from the clamp when a travel command is pending.
- The following is always valid for the axis:
  
  | DB31, ... DBX2.1 (controller enable) = 0: Axis is clamped. |
  | DB31, ... DBX2.1 (controller enable) = 1: Axis is not clamped. |

The following figure shows an example of the interface signals and states upon releasing of the axis clamp:

![Diagram](image)

Figure 3-2 Release axis clamp if MD36052 $MA_STOP_ON_CLAMPING = 'H01'

The part program blocks N310 and N410 refer to the following programming example:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 G0 X0 Y0 Z0 A0 G90 G54 F500</td>
<td></td>
</tr>
<tr>
<td>N101 G641 ADIS=1 ADISPOS=5</td>
<td></td>
</tr>
<tr>
<td>N210 G1 X10</td>
<td>; Machining</td>
</tr>
<tr>
<td>N220 G1 X5 Y20</td>
<td></td>
</tr>
<tr>
<td>N310 G0 Z50</td>
<td>; Retract</td>
</tr>
<tr>
<td>N410 G0 A90</td>
<td>; Turn rotary table</td>
</tr>
<tr>
<td>N510 G0 X100</td>
<td>; Approach</td>
</tr>
</tbody>
</table>
### Optimized release of the axis clamping via travel command

If a clamped axis is to be traversed in continuous-path mode, a travel command is issued for the clamped axis in the rapid traverse blocks (G0) immediately before the traversing block of the clamped axis. This way, the PLC user program can release the axis clamp again in time.

**Note**

The travel command is set to a maximum of two rapid traverse blocks prior (including intermediate blocks) to retain the reference to the initiating part program block.

**Parameterization:**

MD36052 $MA_STOP_ON_CLAMPING = 'H03' (special function for clamped axis)

### Requirements for the PLC user program

- The axis is removed from the clamp as soon as a travel command is pending.
- The axis may be removed from the clamping even when only positioning is being performed (G0).

The following figure shows an example of the interface signals and states upon releasing of the axis clamp:
Automatic stop to set the clamping

If an axis is to be clamped in continuous-path mode, the NC stops the path motion before the next "Non-rapid traverse block" if the axis has not been clamped by then, i.e. the PLC has set the feedrate override value to zero.

Parameterization:

MD36052 $MA_STOP_ON_CLAMPING = 'H04' (special function for clamped axis)

Requirements for the PLC user program

- The axis is always clamped when no travel command is pending.
- The axis does not have to be clamped during positioning of the other axes. It can be seen whether the axes are being positioned depending on whether rapid traverse (G0) is programmed. The stop command is therefore not set immediately at the beginning of the block containing the axis, but rather at the beginning of the next machining block (traversing block that is not traversed with rapid traverse).
- The axis is clamped if the feedrate override of a machining block is not equal to 0. If the axis is clamped before the next machining block, i.e. the feedrate override is other than 0 again, no stop is generated.

Figure 3-3  Release axis clamp if MD36052 $MA_STOP_ON_CLAMPING = 'H03'
The following figure shows an example of the interface signals and states upon setting of the axis clamp. The part program blocks N410, N510, N520 and N610 refer to the schematic example under supplementary conditions.

Figure 3-4  Set axis clamp if MD36052 $MA_STOP_ON_CLAMPING[.] = 'H04'

**Supplementary conditions**

**Continuous-path mode**

For the above-mentioned functions:

- Automatic stop to release the clamping
- Optimized release of the axis clamping via travel command
- Automatic stop to set the clamping

the "Look Ahead" function must be active.
Part program blocks without path motion (e.g. M82/M83) interrupt continuous-path mode and thus also the "Look Ahead" function.

Example:

The part program blocks N320 and N420 are inserted in the programming example used.

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 G0 X0 Y0 Z0 A0 G90 G54 F500</td>
<td></td>
</tr>
<tr>
<td>N101 G64 ADIS=.1 ADISPOS=5</td>
<td></td>
</tr>
<tr>
<td>N210 G1 X10</td>
<td>Machining</td>
</tr>
<tr>
<td>N220 G1 X5 Y20</td>
<td></td>
</tr>
<tr>
<td>N310 G0 Z50</td>
<td>Retract</td>
</tr>
<tr>
<td>N320 M82</td>
<td>No path motion</td>
</tr>
<tr>
<td>N410 G0 A90</td>
<td>Turn rotary table</td>
</tr>
<tr>
<td>N420 M83</td>
<td>No path motion</td>
</tr>
<tr>
<td>N510 G0 X100</td>
<td>Approach</td>
</tr>
<tr>
<td>N520 G0 Z2</td>
<td></td>
</tr>
<tr>
<td>N610 G1 Z-4</td>
<td>Machining</td>
</tr>
<tr>
<td>N620 G1 X0 Y-20</td>
<td></td>
</tr>
</tbody>
</table>

The function behaves as follows:

- **MD36052 $MA_STOP_ON_CLAMPING = 'H03'**
  No longer has an effect.
  The travel command is set in Look Ahead mode only for blocks with active continuous-path mode. M82 generates a stop and thus interrupts the continuous-path mode. The Look Ahead stopping on N410 would not be necessary because stopping occurs anyway.

- **MD36052 $MA_STOP_ON_CLAMPING = 'H04'**
  Generates a stop irrespective of M83 which is executed as a function of "feedrate override 0%". The axis is thus stopped before the first machining block.

**Note**

**MD36052 $MA_STOP_ON_CLAMPING = 'H01' or 'H04'**

Both functions can be used irrespective of the clamping of axes:

- **MD36052 $MA_STOP_ON_CLAMPING = 'H01'**
  Generates a Look Ahead stop for the path motion if no servo enable signal is active for the relevant axis.

- **MD36052 $MA_STOP_ON_CLAMPING = 'H04'**
  Generates a Look Ahead stop for the path motion if the feedrate override = 0% at the transition from the part program blocks with rapid traverse to part program blocks without rapid traverse.

Both functions ensure that the path motion in continuous-path mode is already stopped before the start of the relevant part program block and not just within the part program block.
Block change criterion: Clamping tolerance

After activation of clamping monitoring (DB31, ... DBX2.3 = 1), the block change criterion for traversing blocks in which the axis stops at the end of the block no longer acts as the corresponding exact-stop condition, but rather as the configured clamping tolerance:

MD36050 $MA_CLAMP_POS_TOL (clamping tolerance with interface signal "Clamping active")

Behavior upon releasing of the clamp

If the axis was moved by the clamping process, it is returned by the NC to the position setpoint after releasing of the clamp and setting of the servo enable state. Repositioning depends on whether "Follow-up mode" was activated for the axis:

- Without follow-up mode: Repositioning by position controller
- With follow-up mode: Repositioning by interpolator

See also interface signal DB31, ... DBX1.4 (follow-up mode).

Note

The following interface signals can be evaluated by the PLC user program as the criterion for activation of the "Follow-up mode":

DB31, ... DBX60.6 / 7 (position reached with exact stop coarse/fine)
3.2.3 Speed-setpoint monitoring

Function

The speed setpoint comprises:

- Speed setpoint of the position controller
- Speed setpoint portion of the feedforward control (with active feedforward control only)
- Drift compensation (only for drives with analog setpoint interface)

![Diagram of speed-setpoint monitoring](image)

Figure 3-5 Speed setpoint calculation

The speed-setpoint monitoring ensures by limiting the control or output signal (10 V for analog setpoint interface or rated speed for digital drives) that the physical limitations of the drives are not exceeded:

MD36210 $MA_CTRLOUT_LIMIT$ (maximum speed setpoint)

![Graph of speed setpoint limitation](image)

Figure 3-6 Speed setpoint limitation
3.2 Axis monitoring functions

Speed-setpoint monitoring delay

To prevent an error reaction from occurring in every speed-limitation instance, a delay time can be configured:

MD36220 $MA_CTRLOUT_LIMIT_TIME (speed-setpoint monitoring delay)

Only if the speed limitation is required for longer than the configured time does the corresponding error reaction occur.

Effectiveness

The speed-setpoint monitoring is only active for closed-loop position-controlled axes and cannot be deactivated.

Fault

If the configured delay time is exceeded, the following alarm appears:

25060 "Axis <Axis name> Speed-setpoint limitation"

The affected axis is stopped via the configured braking ramp in follow-up mode:

MD36610 $MA_AX_EMERGENCY_STOP_TIME
(maximum time for braking ramp when an error occurs)

Note

Upon reaching the speed-setpoint monitoring, the position feedback loop of the axis becomes non-linear due to the limitation. Contour errors result if the axis is involved in generating the contour.

3.2.4 Actual-velocity monitoring

Function

The actual-velocity monitoring checks that the actual velocity of a machine axis/spindle does not exceed the configured threshold:

MD36200 $MA_AX_VELO_LIMIT (velocity-monitoring threshold)

The threshold should be 10-15% above the configured maximum velocity.

- For axes:
  MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

- For spindles:
  MD35110 $MA_GEAR_STEP_MAX_VELO_LIMIT[n] (maximum speed of gear stage)

If you use this setting the speed will not normally exceed the velocity-monitoring threshold (exception: Drive error).
**Activation**

The actual-velocity monitoring is activated as soon as the active measuring system returns valid actual values (encoder limit frequency not exceeded).

**Effectiveness**

The actual-velocity monitoring only operates with active position control and the following axis types:
- Linear axes
- Rotary axes
- Open-loop-controlled and position-controlled spindles

**Fault**

If the threshold is exceeded, the following alarm is displayed:

25030 "Axis <Axis name> Actual-velocity alarm limit"

The affected axis is stopped via the configured braking ramp in follow-up mode:

MD36610 $MA_AX_EMERGENCY_STOP_TIME
(maximum time for braking ramp when an error occurs)

### 3.2.5 Measuring system monitoring

The NC has no direct access to the measuring system hardware, therefore measuring system monitoring is mainly performed by the drive software.

**Monitoring functions in the drive**

- Monitoring of hardware faults (e.g. measuring system failure, wire breakage)
- Zero-mark monitoring

**References:**

Drive Functions SINAMICS S120

Measuring system monitoring functions carried out in the drive are mapped on the NCK alarms (alarm 25000 and following) or NC reactions (e.g. abort of referencing or on-the-fly measuring). The exact behavior of the NC depends on the setting in the machine data:
### MD36310 $MA_ENC_ZERO_MONITORING

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>= 0</td>
<td>Monitoring of HW faults: On. If a hardware fault is detected in the active measuring system, POWER ON alarm 25000 is displayed: &quot;Axis &lt;Axis name&gt; Hardware fault active encoder&quot; The affected axis is stopped via the configured braking ramp in follow-up mode: MD36610 $MA_AX_EMERGENCY_STOP_TIME (maximum time for braking ramp when a fault occurs) If a hardware fault is detected in the passive measuring system, alarm 25001 is displayed: &quot;Axis &lt;Axis name&gt; Hardware fault passive encoder&quot; There is no further alarm response.</td>
</tr>
<tr>
<td>Zero-mark monitoring: OFF Alarms 25020 and 25021 (see below) are suppressed.</td>
<td></td>
</tr>
<tr>
<td>= 100</td>
<td>No zero-mark monitoring as well as hiding of all encoder monitoring functions (i.e. in addition to alarm 25020 (25021), alarms 25000 (25001) and 25010 (25011) are suppressed.</td>
</tr>
<tr>
<td>&gt; 0 but &lt; 100</td>
<td>Monitoring of HW faults: ON (see above)</td>
</tr>
<tr>
<td>Zero-mark monitoring: ON If zero-mark monitoring is tripped in the active measuring system, alarm 25020 is displayed: &quot;Axis &lt;Axis name&gt; Zero-mark monitoring active encoder&quot; The affected axis is stopped via the configured braking ramp in follow-up mode: MD36610 $MA_AX_EMERGENCY_STOP_TIME (maximum time for braking ramp when a fault occurs) If zero-mark monitoring is tripped in the passive measuring system, alarm 25021 is displayed: &quot;Axis &lt;Axis name&gt; Zero-mark monitoring passive encoder&quot; There is no further alarm response.</td>
<td></td>
</tr>
<tr>
<td>&gt; 100</td>
<td>Monitoring of HW faults: ON with attenuated error message: The POWER ON alarm 25000 is replaced by the reset alarm 25010 and the reset alarm 25001 replaced by the cancel alarm 25011.</td>
</tr>
<tr>
<td>Zero-mark monitoring: ON (see above)</td>
<td></td>
</tr>
</tbody>
</table>

For details on the alarms, see:

**References:**
- Diagnostics Manual

**Note**

For hardware faults, the referencing status of the machine axis is reset:

DB31, ... DBX60.4/5 (referenced/synchronized 1/2) = 0
Monitoring functions in the NCK

- Encoder-limit-frequency monitoring
- Plausibility check for absolute encoders

3.2.5.1 Encoder-limit-frequency monitoring

Function

The NC encoder-limit-frequency monitoring is based on the configuration and telegram information of the drive. It monitors that the encoder frequency does not exceed the configured encoder limit frequency:

MD36300 $MA_ENC_FREQ_LIMIT (encoder limit frequency)

Encoder-limit-frequency monitoring always refers to the active measuring system selected in the NC/PLC interface:

DB31, ... DBX1.5/1.6 (position measuring system 1/2)

Effectiveness

The encoder limit frequency is operative for:

- Linear axes
- Rotary axes
- Open-loop-controlled and position-controlled spindles
Fault

Upon exceeding of the encoder limit frequency, the following occurs:

- **Message to the PLC:**
  DB31, ... DBX60.2 or 60.3 = 1 (encoder limit frequency exceeded 1 or 2)

- **Spindles**
  Spindles are not stopped but continue to turn with speed control.
  If the spindle speed is reduced so much that the encoder frequency passes below the encoder limit frequency, the actual value system of the spindle is automatically resynchronized.

- **Axes**
  The following alarm is displayed:
  21610 "Channel <Channel number> Axis <Axis name> Encoder <Encoder number> Frequency exceeded"
  The affected axis is stopped via the configured braking ramp in follow-up mode:
  MD36610 $MA_AX_EMERGENCY_STOP_TIME
  (maximum time for braking ramp when an error occurs)

**Note**

If the encoder limit frequency is exceeded, a position-controlled machine axis must be re-referenced (see Section "R1: Referencing (Page 1149)").

3.2.5.2 **Plausibility check for absolute encoders**

**Function**

With absolute encoders (MD30240 $MA_ENC_TYPE = 4), absolute values supplied by the measuring system are used to check the plausibility of the actual value.

During the check, the NC compares the cyclic position value held in the position control cycle clock based on the incremental information from the encoder with a new position value generated directly from the absolute and incremental information and checks that the calculated position difference does not exceed the permissible deviation.

MD36310 $MA_ENC_ZERO_MONITORING (permissible deviation in 1/2 coarse increments between the absolute and the incremental encoder track)

**Note**

The plausibility check of absolute encoders specifically detects all deviations caused by dirt on the absolute track or by faults when transferring the absolute value. However, small errors in the incremental track (burst interference, impulse errors) are not detected. In such instances the plausibility check only responds to deviations in the millimeter range. This form of monitoring should therefore serve as additional monitoring to assist the diagnosis of absolute-position faults.
Note
Rotary absolute encoders

If the plausibility check is to be used for a rotary absolute encoder, the SINAMICS parameter p0979 must be taken into account when setting the modulo range (MD34220 $MA_ENC_ABS_TURNS_MODULO).

Note
Upgrading the NCK software

If the plausibility check is activated in absolute encoders (MD36310 > 0), the existing MD36310 settings must be checked and, if necessary, increased during an upgrade of the NCK software.

Zero mark diagnostics

With absolute encoders, the permissible deviation must be determined for the plausibility check during commissioning. This can be performed via the machine data:

MD36312 $MA_ENC_ABS_ZEROMON_WARNING (zero-mark monitoring warning threshold)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No zero mark diagnostics</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>Permissible deviation in 1/2 coarse increments between the absolute and the incremental encoder track</td>
</tr>
</tbody>
</table>

Procedure when commissioning the system:

1. Deactivate zero-mark monitoring:
   MD36310 $MA_ENC_ZERO_MONITORING = 0
2. Activate zero-mark diagnostics:
   MD36312 $MA_ENC_ABS_ZEROMON_WARNING = 1
3. Move axis and monitor system variable $VA_ENC_ZERO_MON_ERR_CNT (number of detected limit value violations).
4. If $VA_ENC_ZERO_MON_ERR_CNT ≠ 0:
   Increase MD36312 value and repeat step 3.
5. If $VA_ENC_ZERO_MON_ERR_CNT = 0 (over a longer period of time!):
   The correct value for MD36310 is located! Apply the value from MD36312 to MD36310 and then set MD36312 to "0".

Note

Depending on the rigidity of the machine (minimal load masses / moments of inertia are optimum) and the controller settings, the control play "oscillates" with varying degrees of intensity. Account must be taken of this by entering machine-specific limit values in MD36310.
3.2 Axis monitoring functions

Fault

Alarm 25020

If the plausibility check is tripped in the active measuring system, alarm 25020 is displayed:
"Axis <Axis name> Zero-mark monitoring active encoder"

The affected axis is stopped via the configured braking ramp in follow-up mode:

MD36610 $MA_AX_EMERGENCY_STOP_TIME
(maximum time for braking ramp when an error occurs)

Alarm 25021

If the plausibility check is tripped in the passive measuring system, alarm 25021 is displayed:
"Axis <Axis name> Zero-mark monitoring passive encoder"

There is no further alarm response.

Note

In the event of a fault, the adjustment of the absolute encoder is lost and the axis is no longer referenced. The absolute encoder must be readjusted (see Section "Referencing with absolute encoders (Page 1179)").

Note

Errors in the incremental track that cannot be detected with amplitude monitoring can cause position deviations in the millimeter range. The deviation depends on the lattice pitch/line count and the traversing velocity of the axis when the error occurs.

Complete position monitoring is only possible through redundancy, i.e. through comparison with an independent second measuring system.

3.2.5.3 Customized error reactions

Customized zero-mark monitoring

The default alarm and reaction behavior of the zero-mark monitoring can be adapted in absolute measuring systems (MD30240 $MA_ENC_TYPE = 4) with the aid of system variables. This allows you to perform your own monitoring using a synchronized action or OEM application and to use all of the reaction options available in this application, e.g.:

- Transmit alarm
- Use cycles (e.g. approach tool-change position)
- ...

Example:

Users can adjust the alarm and reaction behavior so that when machining an expensive workpiece, which could be damaged if the axis is stopped as a result of an alarm, machining
stops before the machining quality of the workpiece is assessed using appropriate synchronized action commands.

**Effectiveness**

Customized monitoring can be activated in parallel to or as an alternative to standard zero-mark monitoring, depending on the setting in machine data:

MD36310 $MA_ENC_ZERO_MONITORING

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>If only user-specific monitoring is to be implemented, the default zero-mark monitoring must be deactivated: MD36310 = 0 and MD36312 = 0</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>Customized monitoring and standard zero-mark monitoring operate in parallel.</td>
</tr>
<tr>
<td>100</td>
<td>All encoder monitoring functions are deactivated.</td>
</tr>
</tbody>
</table>

If both monitoring functions are active (MD36310 > 0), you can perform **cascaded monitoring**.

Example:

If a value falls below the threshold specified in MD36310, customized monitoring triggers a prewarning; standard zero-marking monitoring will only detect a fault if the threshold is exceeded and will then deactivate automatically.
System variables

You can implement customized error reactions using the following system variables:

<table>
<thead>
<tr>
<th>System variable</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| $VA\_ENC\_ZERO\_MON\_ERR\_CNT[<n>,<axis>] | Number of detected limit value violations. Contains the current number of detected limit value violations when comparing the absolute and the incremental encoder tracks. The value is reset to 0 at:  
  - POWER ON  
  - Selection/deselection of parking  
  Reset does not cause a reset. |
| $VA\_ABSOLUTE\_ENC\_DELTA\_INIT[<n>,<axis>] | Initial difference for absolute encoders. Contains the initial difference between the last buffered absolute position in the static NC memory and the current absolute position. Format of the difference value: Number of internal increments (see MD10200 $MN\_INT\_INCR\_PER\_MM or MD10210 $MN\_INT\_INCR\_PER\_DEG)  
  The value is updated at:  
  - POWER ON  
  - Warm restart  
  - Deselection of parking  
  - Return below the encoder limit frequency  
  There is no reset at reset. |

<n>: Encoder number  
<axis>: Axis name
3.2.6 Limit-switch monitoring

Overview of the end stops and possible limit-switch monitoring:

3.2.6.1 Hardware limit switch

Function

A hardware limit switch is normally installed at the end of the traversing range of a machine axis. It serves to protect against accidental overtravelling of the maximum traversing range of the machine axis while the machine axis is not yet referenced.

If the hardware limit switch is triggered, the PLC user program created by the machine manufacturer sets the corresponding interface signal:

DB31, ... DBX12.0/1 = 1 (hardware limit switch minus/plus)

Parameterization

The braking behavior of the machine axis upon reaching the hardware limit switch is configurable via the machine data:

MD36600 $MA_BRAKE_MODE_CHOICE (braking behavior on hardware limit switch)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Braking with the configured axial acceleration</td>
</tr>
<tr>
<td>1</td>
<td>Rapid stop (set velocity = 0)</td>
</tr>
</tbody>
</table>

Effectiveness

The hardware limit-switch monitoring is active after the controller has ramped up in all modes.
Effect

Upon reaching the hardware limit switch, the following occurs:

- Alarm 21614 "Channel <Channel number> Axis <Axis name> Hardware limit switch <Direction>"
- The machine axis is braked according to the configured braking behavior.
- If the axis/spindle is involved in interpolation with other axes/spindles, these are also braked according to their configured braking behavior.
- The traversing keys of the affected machine axis are blocked based on the direction.

3.2.6.2 Software limit switch

Function

Software limit switches serve to limit the traversing range of a machine axis. Per machine axis and per traversing direction, two (1st and 2nd) software limit switches are available:

- MD36100 POS_LIMIT_MINUS (1st software limit switch minus)
- MD36110 POS_LIMIT_PLUS (1st software limit switch plus)
- MD36120 POS_LIMIT_MINUS2 (2nd software limit switch minus)
- MD36130 POS_LIMIT_PLUS2 (2nd software limit switch plus)

By default, the 1st software limit switch is active. The 2nd software limit switch can be activated for a specific direction with the PLC user program:

- DB31, ... DBX12.2 / 12.3 (2nd software limit switch minus/plus)

Effectiveness

The software limit switches are active:

- Immediately after the successful referencing of the machine axis.
- In all operating modes.

Boundary conditions

- The software limit switches refer to the machine coordinate system.
- The software limit switches must be inside the range of the hardware limit switches.
- The machine axis can be moved to the position of the active software limit switch.
- **PRESET**
  After use of the function **PRESET**, the software limit-switch monitoring is no longer active. The machine must first be re-referenced.
- Endlessly rotating rotary axes
  No software limit-switch monitoring takes place for endlessly rotating rotary axes:
  - MD30310 $MA_ROT_IS_MODULO == 1 (modulo conversion for rotary axis and spindle)
  - Exception: Setup-rotary axes
Effects

Automatic operating modes (AUTOMATIC, MDA)

- Without transformation, without overlaid motion, unchanged software limit switch:
  A part program block with a programmed traversing motion that would lead to overrunning of the software limit switch is not started.

- With transformation:
  Different reactions occur depending on the transformation type:
  - Behavior as above.
  - The part program block with a programmed traversing motion that would lead to overrunning of the software limit switch is started. The affected machine axis stops at the active software limit switch. The other machine axes participating in the traversing motion are braked. The programmed contour is left during this process.

- With overlaid motion
  The part program block with a programmed traversing motion that would lead to overrunning of the software limit switch is started. Machine axes that are traveling with overlaid motion or have traveled with overlaid motion stop at the active software limit switch in question. The other machine axes participating in the traversing motion are braked. The programmed contour is left during this process.

Manual operating modes

- JOG without transformation
  The machine axis stops at the software limit switch position.

- JOG with transformation
  The machine axis stops at the software limit switch position. Other machine axes participating in the traversing motion are braked. The preset path is left during this process.

General

- Changing of the software limit switch (1st ↔ 2nd software limit switch)
  If the actual position of the machine axis after changing lies behind the software limit switch, it is stopped with the maximum permissible acceleration.

- Overrunning the software limit switch in JOG mode
  If the position of the software limit switch is reached and renewed pressing of the traversing button should cause further travel in this direction, an alarm is displayed and the axis is not traversed farther:
  Alarm 10621 "Channel <Channel number> Axis <Axis name> is at the software limit switch <Direction>"
3.2.7 Monitoring of the working area limitation

3.2.7.1 General

Function

The "working area limitation" function can be used to limit the traversing range of a channel's geometry and special axes to a permissible operating range. The function monitors compliance with working area limits both in AUTOMATIC mode and in JOG mode.

The following versions are available:

- Working area limitation in the Basic Coordinate System (BCS)
  The traversing range limits are specified relative to the Basic Coordinate System.

- Working area limitation in the workpiece coordinate system (WCS) or adjustable zero system (AZS)
  The traversing range limits are specified relative to the workpiece coordinate system or to the adjustable zero system.

The two types of monitoring are independent of each other. If they are both active at the same time, the traversing range limit which most restricts the access will take effect, depending on the direction of travel.
Reference point at the tool

Taking into account the tool data (tool length and tool radius) and therefore the reference point at the tool when monitoring the working area limitation depends on the status of the transformation in the channel:

- **Transformation inactive**
  Without transformations during traversing motion with an active tool the position of the tool tip P is monitored, i.e. during the monitoring the tool length is considered automatically. Consideration of the tool radius must be activated separately:
  MD21020 $MC_WORKAREA_WITH_TOOL_RADIUS (Consideration of the tool radius in the working area limitation)

- **Transformation active**
  In the case of certain transformations the monitoring of the working area limitation may differ from the behavior without transformation:
  - The tool length is a component of the transformation ($MC_TRAFO_INCLUDES_TOOL_X = TRUE$): In this case the tool length is not considered, i.e. the monitoring refers to the tool carrier reference point.
  - Transformation with change in orientation: In the case of transformations with changes in orientation, monitoring is always based on the tool center point. MD21020 has no influence.

**Note**

The machine data $MC_TRAFO_INCLUDES_TOOL_... is analyzed only in certain transformations. Condition for a possible evaluation is that the orientation of the tool with respect to the base coordinate system cannot be changed by the transformation. With standard transformations, the condition is only fulfilled for the "inclined axis" type of transformation.

Response

**Automatic operating modes**

- With / without transformation
  The parts program block with a programmed traversing motion that would lead to overrunning of the working area limits is not executed.

- With superimposed motion
  The axis, which would violate the working area limitation due to a superimposed motion, is braked with maximum acceleration and without jerk limits (**BRISK**), and will come to a stop in the position of the working area limitation. Other axes involved in the movement are braked according to current acceleration behavior (e.g. **SOFT**). The path correlation may be lost due to different braking accelerations (contour violation).

**Manual operating modes**

- JOG with / without transformation
  The axis is positioned at the working area limitation and then stopped.
Powerup response

If an axis moves outside the permissible working area when activating the working area limits, it will be immediately stopped with the maximum permissible acceleration.

Overrunning of the working area limitation in JOG mode

In JOG mode, an axis is moved to no further than its working area limit by the control system. When the traverse button is pressed again, an alarm is displayed and the axis does not traverse any further.

Geo-axis replacement

Through the following machine data it is adjustable, whether during geometry axis change the active working area limitation is retained or deactivated:

\[ \text{MD10604 $MN\_WALIM\_GEOAX\_CHANGE\_MODE = <value>} \]

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The working area limitation is deactivated during the geometry axis change.</td>
</tr>
<tr>
<td>1</td>
<td>The working area limitation remains activated during the geometry axis change.</td>
</tr>
</tbody>
</table>

3.2.7.2 Working area limitation in BKS

Application

Using the "working area limitation in BCS", the working area of a machine tool is limited so that the surrounding devices (e.g. tool revolver, measuring stations) are protected against damage.

Working area limits

The lower and upper working area limits of each axes are adjusted through setting data or programmed through part program commands:

Working area limitation through setting data

The adjustments are done through the immediately effective axis-specific setting data:

\[ \text{SD43420 $SA\_WORKAREA\_LIMIT\_PLUS (working area limitation plus)} \]
\[ \text{SD43430 $SA\_WORKAREA\_LIMIT\_MINUS (working area limitation minus)} \]

Programmed working area limitation

The programming is done using the G commands:

\[ \text{G25 X... Y... Z... lower working area limitation} \]
\[ \text{G26 X... Y... Z... upper working area limitation} \]
The programmed working area limitation has priority and overwrites the values entered in SD43420 and SD43430.

**Activation/Deactivation**

**Working area limitation through setting data**

The activation/deactivation of the working area limitation for each axis takes place in a direction-specific manner via the immediately effective setting data:

SD43400 $SA_WORKAREA_PLUS_ENABLE (working area limitation active in the positive direction)

SD43410 $SA_WORKAREA_MINUS_ENABLE (working area limitation active in the negative direction)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The working area limitation in positive or negative direction is <strong>switched off</strong>.</td>
</tr>
<tr>
<td>1</td>
<td>The working area limitation in positive or negative direction is <strong>active</strong>.</td>
</tr>
</tbody>
</table>

**Programmed working area limitation**
Activation or deactivation of the overall "working area limitation in the BCS" is arranged via part program commands:

- **WALIMON** Working area limitation ON
- **WALIMOF** Working area limitation OFF

### Changing the working area limitation

**Working area limitation through setting data**

HMI user interface: Operating area “Parameter”

- **Automatic modes:**
  - Changes: Possible only in the RESET state
  - Effectiveness: Immediately

- **Manual operating modes:**
  - Changes: Always possible
  - Effectiveness: At the start of the next traversing motion

**Programmed working area limitation**

The working area limitation can be changed in the part program via `G25` or `G26 <Axis name> <value>`. The change takes effect immediately.

The new working area limitation value is retained after and NC RESET and POWER ON if the back-up process has been activated in the NCK’s retentive data storage for SD43420 and SD43430:

- MD10710 $MN_PROG_SD_RESET_SAVE_TAB[0] = 43420
- MD10710 $MN_PROG_SD_RESET_SAVE_TAB[1] = 43430

### Reset position

The reset position for the working area limitation (WALIMON or WALIMOF) is configurable via:

- MD20150 $MC_GCODE_RESET_VALUES (RESET position of G groups)

### 3.2.7.3 Working area limitation in WCS/SZS

**Application**

The "working area limitation" in the WCS/SZS enables a channel-specific, flexible workpiece-specific limitation of the traversing range of the channel axes in the workpiece coordinate system (WCS) or settable zero system (SZS). It is intended mainly for use in conventional lathes.
Requirement

The channel axes must be referenced.

Working area limitation group

In order that the axis-specific working area limits do not have to be rewritten for all channel axes when switching axis assignments, e.g. when switching transformations or the active frame on/off, working area limitation groups are available.

A working area limitation group comprises the following data:

- Working area limits for all channel axes
- Reference system of the working area limitation

The number of the working area limitation groups is set in the machine data:

MD28600 $MC_MM_NUM_WORKAREA_CS_GROUPS

Set working area limits

The working area limits are set channel-specifically for each channel axis via the following system variables:

- $P\_WORKAREA\_CS\_LIMIT\_PLUS[<group>, <channel axis name>]
- $P\_WORKAREA\_CS\_LIMIT\_MINUS[<group>, <channel axis name>]

With <group> = number of the working area limitation group

Enable working area limits

The working area limits are enabled channel-specifically for each channel axis via the following system variables:

- $P\_WORKAREA\_CS\_PLUS\_ENABLE[<group>, <channel axis name>]
- $P\_WORKAREA\_CS\_MINUS\_ENABLE[<group>, <channel axis name>]

With <group> = number of the working area limitation group

There is no activation through the enable.

Select reference system

The reference system for a working area limitation group is set via the following system variable for each specific channel:

$P\_WORKAREA\_CS\_COORD\_SYSTEM[<group>] = <value>

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Working area limitation based on the workpiece coordinate system</td>
</tr>
<tr>
<td>3</td>
<td>Working area limitation based on the AZS</td>
</tr>
</tbody>
</table>

With <group> = number of the working area limitation group
Activation of working area limits

The working area limits of a working area limitation group are activated in the part program with the command:

\[ \text{WALCS}n \] (activation of the working area limits of group \( n \), where \( n = 1, 2, \ldots \) )

Deactivation of working area limits

The working area limits of a working area limitation group are deactivated in the part program with the command:

\[ \text{WALCS}0 \] (deactivation of the working area limits active in the channel)

Change working area limits

The working area limits can be changed at any time via the system variables mentioned above. Changes take effect with the next activation of the working area limitation group (WALCS\( n \)).

Data storage

The system variables of the working area limits are stored retentively in the static memory of the NC.

Note

For the storage of the limiting values for the linear axes, the default setting is considered for the system of units (\$MN_SCALING_SYSTEM_IS_METRIC).

Data backup

The system variables of the working area limits can be backed up in separate files:

<table>
<thead>
<tr>
<th>Backup file</th>
<th>For the backup of:</th>
</tr>
</thead>
<tbody>
<tr>
<td>_N_CHx_WAL</td>
<td>Values of the system variables for the channel ( x ).</td>
</tr>
<tr>
<td>_N_COMPLETE_WAL</td>
<td>Values of the system variables for all channels.</td>
</tr>
</tbody>
</table>

Note

The system variables of the working area limits are part of the "_N_INITIAL_INI" file.

Behavior in JOG mode

Initial situation:

- In JOG mode, several geometry axes traverse simultaneously (e.g. using several handwheels)
- A rotating frame is active between the basic coordinate system (BCS) and the reference coordinate system of the working area limitation (WCS or SZS)
Behavior when a working area limitation responds:

- The traversing motions of the geometry axes that are not affected are continued
- The affected geometry axis is stopped at the working area limit

Set initial setting

The specification of the working area limitation group that is to take effect at power up, reset or end of the part program and start of the part program is performed channel-specifically via the machine data:

\[ \text{MD20150 } \$MC\_GCODE\_RESET\_VALUE[59] = \langle \text{group} \rangle \]

The working area limitation group that is to take effect is still dependent on the setting in the machine data:

\[ \text{MD20152 } \$MC\_GCODE\_RESET\_MODE[59] = \langle \text{mode} \rangle \]

<table>
<thead>
<tr>
<th>&lt;mode&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The working area group takes effect in accordance with MD20150</td>
</tr>
<tr>
<td>1</td>
<td>The last active working area group remains active</td>
</tr>
</tbody>
</table>

3.2.8 Parking a machine axis

If a machine axis is brought into the"Parking" state, then for this particular axis, no encoder actual values are acquired, and all of the monitoring functions described in the preceding sections (measuring system, standstill, clamping monitoring, etc.) are deactivated.

Activation / deactivation

Activation

"Parking" a machine axis activated by resetting the axis-specific NC/PLC interface signals for the parameterized position measuring systems and the controller enable of the machine axis:

- DB31, ... DBX1.5 (position measuring system 1) = 0
- DB31, ... DBX1.6 (position measuring system 2) = 0
- DB31, ... DBX2.1 (controller enable) = 0

The encoder status of the position measuring system of the axis is then set to "Not referenced":

- DB31, ... DBX60.4 (referenced/synchronized, position measuring system 1) == 0
- DB31, ... DBX60.5 (referenced/synchronized, position measuring system 2) == 0

The following further NC/PLC interface signals are also reset:

- DB31, ... DBX61.5 (position controller active) == 0
- DB31, ... DBX61.6 (speed controller active) == 0
- DB31, ... DBX61.7 (current controller active) == 0
- DB31, ... DBX93.7 (pulses enabled) == 0
Deactivation

"Parking" a machine axis is deactivated by setting the axis-specific NC/PLC interface signals for the position measuring system to be activated and the controller enable of the machine axis:

- DB31, ... DBX1.5 (position measuring system 1) = 1
- or
- DB31, ... DBX1.6 (position measuring system 2) = 1
- DB31, ... DBX2.1 (controller enable) = 1

The position control for the machine axis becomes active again at the current position.

The encoder state of the position measuring systems depends on the measuring system type:

- Incremental position measuring system ⇒ "not referenced" state
  - DB31, ... DBX60.4 (referenced/synchronized, position measuring system 1) == 0
  - DB31, ... DBX60.5 (referenced/synchronized, position measuring system 2) == 0
- Absolute position measuring system ⇒ "referenced/synchronized" state
  - DB31, ... DBX60.4 (referenced/synchronized, position measuring system 1) == 1
  - DB31, ... DBX60.5 (referenced/synchronized, position measuring system 2) == 1

The following NC/PLC interface signals are also set again:

- DB31, ... DBX61.5 (position controller active) == 1
- DB31, ... DBX61.6 (speed controller active) == 1
- DB31, ... DBX61.7 (current controller active) == 1
- DB31, ... DBX93.7 (pulses enabled) == 1
- DB31, ... DBX102.5 (position measuring system 1 activated) == 1
- DB31, ... DBX102.6 (position measuring system 2 activated) == 1

Incremental position measuring systems

After deactivation of the "parking" state, incremental position measuring systems have to be referenced before they have "referenced" encoder status.

⚠️ WARNING

Incorrect synchronization of the position measuring system caused by offset of the actual machine axis position

If changes have been made to the position measuring system during "parking" that require a change to the parameterized machine data, for example, another encoder has been mounted, the position measuring system must be completely remeasured and referenced. See Section "R1: Referencing (Page 1149)."
Machine axis without position measuring system

For a machine axis without a position measuring system (speed-controlled spindle), a status equivalent to "parking" is activated by canceling the controller enable:

- DB31, ... DBX2.1 (controller enable) = 0

3.2.9 Parking the passive position measuring system

3.2.9.1 Function

Contrary to the "Parking a machine axis" function, for which all position measuring systems of a machine axis are deactivated, using the "Park the passive position measuring system" function, users have the option, of only "parking" the passive position measuring system of a machine axis (i.e. to deactivate the encoder evaluation and monitoring in the drive and in the control), while the active position measuring system can remain operational.

Note

For an explanation of active/passive measuring system, refer to "Setpoint/actual-value system (Page 336)".

Application

The "Park passive measuring system" function can, for example, be used in the following cases:

- Changing attachment heads with and without integrated encoder

  Using the "Park passive position measuring system" function, it is possible to mount attachment heads alternating with and without integrated encoder for different machining tasks on the main spindle, without the missing encoder signals initiating drive and control faults.

  See also:

  - Example: Changing an attachment head for a direct position measuring system (Page 119)
  - Example: Changing an attachment head for two direct position measuring systems (Page 123)

- Using linear position measuring systems, which are not available over the complete traversing range of a machine axis

  Using the "Park passive position measuring system" function, it is possible to pass through the range outside the linear position measuring system, without the missing encoder signals initiating drive and control faults.

  See also:

  - "Example: Measuring system switchover when encoders are missing in certain parts of the range (Page 127)".
Activation / deactivation

Activation
The passive position measuring system of a machine axis is parked under the following conditions:

- "Park passive position measuring system" function is active for the measuring system:
  \[ MD31046 \$MA\_ENC\_PASSIVE\_PARKING[<n>] = 1 \]
  with \(<n> = 0\) (position measuring system 1) or \(<n> = 1\) (position measuring system 2)

**Note**
MD31046 is **not active**:
- for axes with fewer than two encoders:
  \[ MD30200 \$MA\_NUM\_ENCS < 2 \]
- for simulated encoders:
  \[ MD30240 \$MA\_ENC\_TYPE = 0 \]

**Note**
For position measuring systems, which are used as motor measuring systems, the "Park passive position measuring system" function should be deactivated (MD31046 = 0)!

and

- The user sets the following NC/PLC interface signal to "0":
  \[ DB31, ... DBX1.5 \text{ (position measuring system 1)} = 0 \]
  or
  \[ DB31, ... DBX1.6 \text{ (position measuring system 2)} = 0 \]

The controller then sets the status for the activation state of the position measuring system to "0":

\[ DB31, ... DBX102.5 \text{ (position measuring system 1 activated)} == 0 \]

or

\[ DB31, ... DBX102.6 \text{ (position measuring system 2 activated)} == 0 \]

The position measuring system is now no longer monitored and updated.

Deactivation
"Park" is deactivated if the user activates the position measuring system:

\[ DB31, ... DBX1.5 \text{ (position measuring system 1)} = 1 \]

or

\[ DB31, ... DBX1.6 \text{ (position measuring system 2)} = 1 \]

The controller then sets the status for the activation state of the position measuring system back to "1":

\[ DB31, ... DBX102.5 \text{ (position measuring system 1 activated)} == 1 \]

or
DB31, ... DBX102.6 (position measuring system 2 activated) == 1

Note
Switching over to a parked position measuring system takes longer than to a non-parked position measuring system. Because of the time taken, we recommend switching over while the axes are stationary.

Position of the position measuring system

Absolute position measuring systems
For absolute position measuring systems, the position after deactivating "park" corresponds to the actual absolute encoder position.

The position measuring system is referenced:
DB31, ... DBX60.4 (referenced/synchronized, position measuring system 1) == 1
or
DB31, ... DBX60.5 (referenced/synchronized, position measuring system 2) == 1

Incremental position measuring systems
For incremental position measuring systems, the position after deactivating "park" always corresponds to the last deactivation position of the position measuring system.

Switching over to the parked position measuring system is only realized if the parameterized permissible deviation between the actual values of the two position measuring systems (see MD36500 $MA_ENC_CHANGE_TOL) is not exceeded. Otherwise, users must apply the "Park a machine axis" function, in which case such a check is not made.

The position measuring system is not referenced:
DB31, ... DBX60.4 (referenced/synchronized, position measuring system 1) == 0
or
DB31, ... DBX60.5 (referenced/synchronized, position measuring system 2) == 0

Incremental position measuring systems with position transfer
Alternatively, for incremental position measuring systems, when the "Park passive position measuring system" is active (MD31046 $MA_ENC_PASSIVE_PARKING[<n>] = 1) it is possible, after deactivating "Park" to transfer the position, and where relevant, also the "Referenced" status, from the previously active position measuring system.

For every position measuring system of a machine axis, this function can be activated using machine data:
MD34210 $MA_ENC_REFP_STATE[<n>]
with \(<n> = 0\) (position measuring system 1) or \(1\) (position measuring system 2)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Only the position from the previous active position measuring system is transferred. The position measuring system is not referenced: DB31, ... DBX60.4 (referenced/synchronized, position measuring system 1) == 0 or DB31, ... DBX60.5 (referenced/synchronized, position measuring system 2) == 0</td>
</tr>
<tr>
<td>2</td>
<td>Position and &quot;Referenced&quot; status are transferred from the previously active position measuring system. The position measuring system is referenced: DB31, ... DBX60.4 (referenced/synchronized, position measuring system 1) == 1 or DB31, ... DBX60.5 (referenced/synchronized, position measuring system 2) == 1</td>
</tr>
</tbody>
</table>

**Note**

Position and "Referenced" status of the previously active position measuring system are only transferred for incremental position measuring systems, depending on MD34210 $MA\_ENC\_REFP\_STATE[<n>\] – and only when the "Park passive position measuring system" function is active (MD31046 $MA\_ENC\_PASSIVE\_PARKING[<n>] = 1).

The transferred position has the accuracy of the previous active position measuring system. The position measuring system should be rereferenced if this accuracy is not adequate.

**WARNING**

Incorrect synchronization of the position measuring system caused by offset of the actual machine axis position

If changes have been made to the position measuring system during "parking" that require a change to the parameterized machine data, for example, another encoder has been mounted, the position measuring system must be completely remeasured and referenced. See Chapter "R1: Referencing (Page 1149)".

### 3.2.9.2 Supplementary conditions

**Interaction with "dual position feedback"**

The "Park passive position measuring system" function cannot be used in conjunction with the "dual position feedback" function (MD32960 $MA\_POSCTRL\_DUAL\_FEEDBACK\_TIME > 0).

**Interaction with "position difference input"**

The "Park passive position measuring system" function cannot be used in conjunction with the "position difference input" function (MD32950 $MA\_POSCTRL\_DAMPING > 0).
Interaction with APC (option for SINUMERIK 840D sl)

The "Park passive measuring system" function cannot be used in conjunction with the "Advanced positioning control (APC)" drive function.

Interaction with encoder safety protection concept

Only the 1 encoder safety protection concept can be used in conjunction with the "Park passive position measuring system" function.

Interaction when connecting and disconnecting at the DRIVE-CLiQ

If, instead of the encoder cable, the DRIVE-CLiQ cables, e.g. between the SMC and Motor Module are unplugged and plugged, such encoders can only be unparked without fault via the "Parking a machine axis" (Page 113)" function.

3.2.9.3 Example: Changing an attachment head for a direct position measuring system

Initial situation

- Attachment head "A" has an encoder E2.
- Attachment head "B" does not have an encoder.
- Spindle "SP" has an encoder E1.
- One of the following two telegram types is configured in MD13060 $MN_DRIVE_TELEGRAM_TYPE (standard telegram type for PROFIdrive):
  - Telegram 116 (motor encoder + an external encoder)
  - Telegram 136 (motor encoder + an external encoder, and torque precontrol)
- The following position measuring systems are configured in the spindle "SP":
  - Motor encoder E1 as position measuring system 1
  - Direct encoder E2 as position measuring system 2
- Attachment head "A" with encoder E2 is currently mounted on the spindle.
- Position measuring system 2 is the active measuring system:
  - $DB31, ... DBX1.6 = 1
  - Position measuring system 1 is passive.
- The "Park passive position measuring system" function is:
  - not active for position measuring system 1:
    - MD31046 $MA_ENC_PASSIVE_PARKING [ 0 ] = 0
  - active for position measuring system 2:
    - MD31046 $MA_ENC_PASSIVE_PARKING [ 1 ] = 1
Objective

The user would like to change from attachment head "A" to attachment head "B."

Implementation
Before changing an attachment head, using the function "Parking a machine axis (Page 113)"; the user must deactivate all position measuring systems of the machine axis:

DB31, ... DBX1.5 (position measuring system 1) = 0
DB31, ... DBX1.6 (position measuring system 2) = 0

The controller then resets the status signals for the position measuring systems:

DB31, ... DBX102.5 (position measuring system 1 activated) == 0
DB31, ... DBX102.6 (position measuring system 2 activated) == 0

The user waits for the status signals and only now removes the attachment head "A" from the spindle. This electrically disconnects the encoder cable between the attachment head "A" and coupling. The absence of encoder E2 does not generate any NC or drive faults.

Attachment head "B" is now mounted on the spindle.

The user only activates position measuring system 1:

DB31, ... DBX1.5 (position measuring system 1) = 1

The control then sets the status signal:

DB31, ... DBX102.5 (position measuring system 1 activated) == 1

The "Park passive position measuring system" function is active for position measuring system 2. This means that position measuring system 2 does not become passive, but remains in the "Park" state.

Objective

The user now wishes to remount attachment head "A" again.
Using the "Park machine axis" function, the user deactivates position measuring system 1:
DB31, ... DBX1.5 (position measuring system 1) = 0
The controller then resets the status signal for the position measuring system:
DB31, ... DBX102.5 (position measuring system 1 activated) == 0

The user waits for the status signal and only now removes the attachment head "B" from the spindle.
Attachment head "A" is now mounted on the spindle.
The user activates position measuring system 2:
DB31, ... DBX1.6 (position measuring system 2) = 1
As a consequence, position measuring system 1 is also simultaneously activated; This is because the "Park passive position measuring system" function is not active for position measuring system 1 (motor measuring system!). Position measuring system 1 becomes a passive position measuring system.
The controller then sets the status signals for the position measuring systems:
DB31, ... DBX102.5 (position measuring system 1 activated) == 1
DB31, ... DBX102.6 (position measuring system 2 activated) == 1

3.2.9.4 Example: Changing an attachment head for two direct position measuring systems

Initial situation
- Attachment head "A" has an encoder E3.
- Attachment head "B" does not have an encoder.
- Spindle "SP" has two encoders E1 and E2.
- One of the following two telegram types is configured in MD13060
  $MN_DRIVE_TELEGRAM_TYPE (standard telegram type for PROFIdrive):
  - Telegram 118 (two external encoders)
  - or
  - Telegram 138 (two external encoders, plus torque precontrol)

References:
For information with regard to the encoder assignment, see:
Commissioning Manual, CNC Commissioning: NC, PLC, Drive;
Section: "Communication between the NC and the drive" > "Drives: Assign axis"
- The following position measuring systems are configured in the spindle "SP":
  - Direct encoder E2 as position measuring system 1
  - Direct encoder E3 as position measuring system 2
- Attachment head "A" with encoder E3 is currently mounted on the spindle.
- Position measuring system 2 is the active measuring system:
  DB31, ... DBX1.6 = 1
  Position measuring system 1 is passive.
- The "Park passive position measuring system" function is:
  - Not active for position measuring system 1:
    MD31046 $MA_ENC_PASSIVE_PARKING [ 0 ] = 0
  - Active for position measuring system 2:
    MD31046 $MA_ENC_PASSIVE_PARKING [ 1 ] = 1
Objective

The user would like to change from attachment head "A" to attachment head "B."

Implementation
Before changing an attachment head, using the function "Parking a machine axis (Page 113)", the user must deactivate all position measuring systems of the machine axis:

DB31, ... DBX1.5 (position measuring system 1) = 0
DB31, ... DBX1.6 (position measuring system 2) = 0

The controller then resets the status signals for the position measuring systems:

DB31, ... DBX102.5 (position measuring system 1 activated) == 0
DB31, ... DBX102.6 (position measuring system 2 activated) == 0

The user waits for the status signals and only now removes the attachment head "A" from the spindle. This electrically disconnects the encoder cable between the attachment head "A" and coupling. The absence of encoder E3 does not generate any NC or drive faults.

Attachment head "B" is now mounted on the spindle.

The user only activates position measuring system 1:

DB31, ... DBX1.5 (position measuring system 1) = 1

The control sets the status signal:

DB31, ... DBX102.5 (position measuring system 1 activated) == 1

The "Park passive position measuring system" function is active for position measuring system 2. This means that position measuring system 2 does not become passive, but remains in the "Park" state.

Objective

The user now wishes to remount attachment head "A" again.
① Using the "Park machine axis" function, the user deactivates position measuring system 1:
\[ DB31, ... DBX1.5 \text{ (position measuring system 1)} = 0 \]
The controller then resets the status signal for the position measuring system:
\[ DB31, ... DBX102.5 \text{ (position measuring system 1 activated)} == 0 \]
② The user waits for the status signal and only now removes the attachment head "B" from the spindle.
Attachment head "A" is now mounted on the spindle.

The user activates position measuring system 2:
DB31, ... DBX1.6 (position measuring system 2) = 1

As a consequence, position measuring system 1 is also simultaneously activated; This is because the "Park passive position measuring system" function is not active for position measuring system 1. Position measuring system 1 becomes a passive position measuring system.

The controller then sets the status signals for the position measuring systems:
DB31, ... DBX102.5 (position measuring system 1 activated) == 1
DB31, ... DBX102.6 (position measuring system 2 activated) == 1

3.2.9.5 Example: Measuring system switchover when encoders are missing in certain parts of the range

In the following example, the direct linear position measuring system is only available in the machining zones, while only the motor measuring system is available outside the machining zones.

Initial situation

- Linear axis "X" has two incremental encoders:
  - Motor encoder E1
  - Direct linear encoder E2
- Direct linear encoder E2 is only available in the machining range.
- One of the following two telegram types is configured in MD13060 $MN_DRIVE_TELEGRAM_TYPE (standard telegram type for PROFIdrive):
  - Telegram 116 (motor encoder + an external encoder)
  - Telegram 136 (motor encoder + an external encoder, and torque precontrol)
- The following position measuring systems are configured for the machine axis:
  - Motor encoder E1 as position measuring system 1
  - Direct linear encoder E2 as position measuring system 2
- When the machine is switched on, the user activates both position measuring systems:
  - DB31, ... DBX1.5 (position measuring system 1) = 1
  - DB31, ... DBX1.6 (position measuring system 2) = 1
- When both position measuring systems are simultaneously activated, then the control selects position measuring system 1 as active position measuring system.
- Position measuring system 2 is selected for machining:
  - DB31, ... DBX1.5 (position measuring system 1) = 0
  - DB31, ... DBX1.6 (position measuring system 2) = 1
The "Park passive position measuring system" function is:

- not active for position measuring system 1:
  MD31046 $MA_ENC_PASSIVE_PARKING [ 0 ] = 0
- active for position measuring system 2:
  MD31046 $MA_ENC_PASSIVE_PARKING [ 1 ] = 1

Transfer of the position and the "referenced" status is activated for position measuring system 2:
MD34210 $MA_ENC_REFP_STATE[ 1 ] = 2

Objective

When passing through the range outside linear position measuring system E2, the missing encoder signals should not initiate any faults in the drive and in the control.
Implementation

A3: Axis Monitoring, Protection Zones
3.2 Axis monitoring functions

Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2 129
Before the table reaches the end of the linear position measuring system, a switchover must be made to the motor measuring system. The user does this by activating both position measuring systems:

DB31, ... DBX1.5 (position measuring system 1) = 1
DB31, ... DBX1.6 (position measuring system 2) = 1

Using the "Park passive position measuring system" function, the linear position measuring system, which is passive after the measuring system switchover, is parked by the control.

The control resets the status signal:

DB31, ... DBX102.6 (position measuring system 2 activated) == 0

The user waits for the status signal before continuing to traverse in the range outside the linear position measuring system.

Traversing through the range outside the linear position measuring system with the motor measuring system.

If the table returns to the linear position measuring system range, at standstill, the user switches from the motor measuring system to the linear position measuring system:

DB31, ... DBX1.5 (position measuring system 1) = 0
DB31, ... DBX1.6 (position measuring system 2) = 1

The control sets the status signal:

DB31, ... DBX102.6 (position measuring system 2 activated) == 1

The motor measuring system becomes a passive position measuring system.

Result

Both position measuring systems are referenced. The position of the linear position measuring system corresponds to the position of the motor measuring system at the switchover instant. The linear position measuring system must be rereferenced if the position accuracy is not sufficient.

3.3 Protection zones

3.3.1 General

Function

Protection zones are static or moveable in 2- or 3-dimensional ranges within a machine to protect machine elements against collisions.
The following elements can be protected:

- Permanent parts of the machine and attachments (e.g. toolholding magazine, swiveling probe). Only the elements that can be reached by possible axis constellations are relevant.
- Moving parts belonging to the tool (e.g. tool, toolholder)
- Moving parts belonging to the workpiece (e.g. parts of the workpiece, clamping table, clamping shoe, spindle chuck, tailstock).

Protection zones are defined via part program instructions or system variables so that they completely surround the element to be protected. The activation and deactivation of protection zones also takes place via part program instructions.

Protection-zone monitoring by the NC is channelspecific, i.e. all the active protection zones of a channel monitor one another for collisions.

**Definition of a protection zone**

It is possible to define 2dimensional or 3dimensional protection zones as polygons with a maximum of ten corner points. The protection zones can also contain arc contour elements.

Polygons are defined in a previously defined plane.

Expansion in the 3rd dimension can be limited between $-\infty$ to $+\infty$.

The following four cases are possible:

- Dimension of the protection zone from $-\infty$ to $+\infty$
- Dimension of the protection zone from $-\infty$ to the upper limit
- Dimension of the protection zone from the lower limit to $+\infty$
- Dimension of protection zone from lower limit to upper limit.

**Coordinate system**

The definition of a protection zone takes place with reference to the geometric axis of a channel in the basic coordinate system.

**Reference**

- Tool-related protection zones
  Coordinates for toolrelated protection zones must be given as absolute values referred to the tool carrier reference point F.

- Workpiece-related protection zones
  Coordinates for workpiece-related protection zones must be given as absolute values referred to the zero point of the basic coordinate system.

**Note**

If no toolrelated protection zone is active, the tool path is checked against the workpiece-related protection zones.

If no workpiece-oriented protection zone is active, protection-zone monitoring does not take place.
Orientation

The orientation of the protection zones is determined by the plane definition (abscissa/ordinate), in which the contour is described, and the axis perpendicular to the contour (vertical axis).

The orientation of the protection zones must be the same for the tool and workpiece-related protection zones.

3.3.2 Types of protection zone

Machine-defined and channel-defined protection zones

- Machine-defined protection zone
  Data for machine-related protection zones are defined once in the control. These protection zones can be activated by all channels.

- Channel-defined protection zones
  Data for channel-related protection zones are defined in a channel. These protection zones can be activated only by this channel.

Example: Doubleslide turning machine

- The tool-related protection zones are assigned to channel 1 or 2.
- The workpiece-related protection zones are assigned to the machine.
- The coordinate system must be identical for both channels.

Maximum number of protection areas

The maximum definable number of machine- and channel-related protection zones is set via:

MD18190 $MN_MM_NUM_PROTECT_AREA_NCK (Number of files for machine-related protection zones)

MD28200 $MC_MM_NUM_PROTECT_AREA_CHAN (Number of files for channel-specific protection zones)

Coordinates

The coordinates of a protection zone must always be programmed as absolute values with respect to the reference point of the protection zone. When the protection zone is activated via the part program it is possible to apply a relative offset to the reference point of the protection zone.
Examples

In the following figures some examples for protection zones have been presented:

Figure 3-8 Example of application on turning machine
3.3 Protection zones

Figure 3-9 Example of a milling machine

Figure 3-10 Example of a turning machine with relative protection zone for tailstock
3.3.3 Definition via part program instruction

General

A protection-zone definition must contain the following information:

- Protection zone type (workpiece- or tool-related)
- Orientation of the protection zone
- Type of limitation in the third dimension
- Upper and lower limits of the protection zone in the third dimension
- Activation type ("Protection zone immediately active": only possible via system variable)
- Contour elements

Definition of protection zones

The following systematics must be maintained in the definition of protection zones:

- Definition of the working plane: G17, G18 or G19
- Definition beginning
  - Channel-specific protection zones: CPROTDEF(...)
  - Machine or NC-specific protection zone: NPROTDEF(...)
- Contour description for protection zone
- End of definition: EXECUTE(...)

Definition of the working plane

The desired working plane to which the contour description of the protection zone refers must be selected with G17, G18, G19 before start of the definition. It may not be changed before the end of the definition. Programming of the applicate is not permitted between start and end of the definition.

Definition beginning

The definition start is defined by the corresponding subroutine:

- CPROTDEF(n, t, applim, appplus, appminus)
- NPROTDEF(n, t, applim, appplus, appminus)

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>INT</td>
<td>Number of defined protection zone</td>
</tr>
<tr>
<td>t</td>
<td>BOOL</td>
<td>Protection zone type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tool-oriented protection zone</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Workpiece-related protection zone</td>
</tr>
</tbody>
</table>
### Contour description for protection zone

The contour of a protection zone is described with traversing motions. These are not executed and have no connection to previous or subsequent geometry descriptions. They only define the protection zone.

The contour of a protection zones is specified with up to eleven traversing movements in the selected working plane. The first traversing movement is the movement to the contour. The last point in the contour description must always coincide with the first point of the contour description. In the case of rotationsymmetrical contours (e.g. spindle chuck), the whole contour must be described (not merely the contour to the turning center).

The valid protection zone is the zone left of the contour:

- **Internal protection zone**
  
  The contour of an internal protection zone must described in the counterclockwise direction.

- **External protection zones (permitted only for workpiece-related protection zones)**
  
  The contour of an external protection zone must be described in the clockwise direction.

![Diagram of protection zones](image)

**Figure 3-11** Examples: External and internal protection zone

Toolrelated protection zones must be convex. If a concave protection zone is required, the protection zone must be divided up into several convex protection zones.
Figure 3-12 Examples: convex and concave tool-related protection zones

Contour elements

The following contour elements are permissible:

- **G0, G1** for straight contour elements
- **G2** for circle segments in the clockwise direction
  Permissible only for workpiece-related protection zones. Not permissible for tool-related protection zones because they must be convex.
- **G3** for circular segments in the counterclockwise direction

A protection zone cannot be described by a complete circle. A complete circle must be divided into two half circles.

The sequence **G2, G3 or G3, G2** is not permitted. A short **G1** block must be inserted between the two circular blocks.

Constraints

During the definition of a protection zone, the following functions must not be active or used:

- Tool radius compensation (cutter radius compensation, tool nose radius compensation)
- Transformation
- Reference point approach (**G74**)
- Fixed point approach (**G75**)
- Dwell time (**G4**)
- Block search stop (**STOPRE**)
- End of program (**M17, M30**)
- M functions: **M0, M1, M2**

Programmable frames (**TRANS, ROT, SCALE, MIRROR**) and configurable frames (**G54 to G57**) are ineffective.
Inch/metric switchovers with G70/G71 or G700/G710 are effective.

End of definition

The end of definition is defined by the following subroutine:

```
EXECUTE (NOT_USED)
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT_USED</td>
<td>INT</td>
<td>Error variable has no effect in protection zones with EXECUTE.</td>
</tr>
</tbody>
</table>

The definition of a machine-specific or channel-specific protection zone is completed with the subroutine EXECUTE(n).

3.3.4 Definition as per system variable

General

When the protection zones are defined via part program instructions (see Section "Definition via part program instruction (Page 135)"), the protection zone data is stored in system variables. The system variables can also be written directly so that the definition of protection areas can also be performed directly in the system variables.

The same supplementary conditions apply for the definition of the contour of a protection zone as for a protection zone definition via part program instructions.

System variables

The protection zone definitions cover following system variables:

<table>
<thead>
<tr>
<th>System variable</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SN_PA_ACTIV_IMMED[n]</td>
<td>BOOL</td>
<td>Activation type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The protection zone is active / not active immediately after the power up of the controller and the referencing of the axes.</td>
</tr>
<tr>
<td>$SC_PA_ACTIV_IMMED[n]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_PA_T_W[n]</td>
<td>INT</td>
<td>Protection zone type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Workpiece-related protection zone</td>
</tr>
<tr>
<td>$SC_PA_T_W[n]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1st and 2nd geometry axis</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3rd and 1st geometry axis</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2nd and 3rd geometry axis</td>
</tr>
<tr>
<td>$SN_PA_ORI[n]</td>
<td>INT</td>
<td>Orientation of the protection zone, i.e. polygon definition in the plane of:</td>
</tr>
<tr>
<td>$SC_PA_ORI[n]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### System variables

<table>
<thead>
<tr>
<th>System variable</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SN_PA_LIM_3DIM[n]$</td>
<td>INT</td>
<td>Type of limitation in the third dimension</td>
</tr>
<tr>
<td>$SC_PA_LIM_3DIM[n]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_PA_PLUS_LIM[n]$</td>
<td>REAL</td>
<td>Value of the limit in the positive direction in the 3rd dimension</td>
</tr>
<tr>
<td>$SC_PA_PLUS_LIM[n]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_PA_MINUS_LIM[n]$</td>
<td>REAL</td>
<td>Value of the limit in the negative direction in the 3rd dimension</td>
</tr>
<tr>
<td>$SC_PA_MINUS_LIM[n]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_PA_CONT_NUM[n]$</td>
<td>INT</td>
<td>Number of valid contour elements</td>
</tr>
<tr>
<td>$SC_PA_CONT_NUM[n]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[n, i]$</td>
<td>INT</td>
<td>Contour type[i], contour type (G1, G2, G3) of the nth contour element</td>
</tr>
<tr>
<td>$SC_PA_CONT_TYP[n, i]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[n, i]$</td>
<td>REAL</td>
<td>End point of the contour[i], abscissa value</td>
</tr>
<tr>
<td>$SC_PA_CONT_ABS[n, i]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[n, i]$</td>
<td>REAL</td>
<td>End point of the contour[i], ordinate value</td>
</tr>
<tr>
<td>$SC_PA_CONT_ORD[n, i]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[n, i]$</td>
<td>REAL</td>
<td>Center point of the circular contour[i], absolute abscissa value</td>
</tr>
<tr>
<td>$SC_PA_CENT_ABS[n, i]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[n, i]$</td>
<td>REAL</td>
<td>Center point of the circular contour[i], absolute ordinate value</td>
</tr>
<tr>
<td>$SC_PA_CENT_ORD[n, i]$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$SN\_...$ are system variables for NC and machine-specific protection zones.
$SC\_...$ are system variables for channel-specific protection zones.

The index "n" corresponds to the number of the protection zone: 0 = 1. Protection zone
The index "i" corresponds to the number of the contour element: 0 = 1. Contour element
The contour elements must be defined in ascending order.

### Note

The system variables of the protection zone definitions are not restored with REORG.

### Data of the protection zone definitions

#### Data storage

The protection zone definitions are stored in the following files:

<table>
<thead>
<tr>
<th>File</th>
<th>Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>_N_NCK_PRO</td>
<td>Data block for NC-specific protection zones</td>
</tr>
<tr>
<td>_N_CHAN1_PRO</td>
<td>Data block for channel-specific protection zones in channel 1</td>
</tr>
<tr>
<td>_N_CHAN2_PRO</td>
<td>Data block for channel-specific protection zones in channel 2</td>
</tr>
</tbody>
</table>

#### Data backup
The protection zone definitions are saved in the following files:

<table>
<thead>
<tr>
<th>File</th>
<th>Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>_N_INITIAL_INI</td>
<td>All data blocks of the protection zones</td>
</tr>
<tr>
<td>_N_COMPLETE_PRO</td>
<td>All data blocks of the protection zones</td>
</tr>
<tr>
<td>_N_CHAN_PRO</td>
<td>All data blocks of the channel-specific protection zones</td>
</tr>
</tbody>
</table>

### 3.3.5 Activating and deactivating protection zones

The activation state of a protection zone can have the following values:

- Activated
- Preactivated
- Preactivated with conditional stop
- Deactivated

#### Activation, preactivation and deactivation via part program

The activation status of a protection zone can be changed in the part program at any time via the following functions:

- Channel-specific protection zone:
  \[ \text{CPROT}(<n>, <state>[, <xMov>, <yMov>, <zMov>]) \]
- Machine-specific protection zone:
  \[ \text{NPROT}(<n>, <state>[, <xMov>, <yMov>, <zMov>]) \]

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;n&gt;</td>
<td>INT</td>
<td>Protection area number</td>
</tr>
<tr>
<td>&lt;state&gt;</td>
<td>INT</td>
<td>Activation status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 Deactivated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 Preactivated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 Activated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 Preactivated with conditional stop</td>
</tr>
<tr>
<td>&lt;xMov&gt;, &lt;yMov&gt;, &lt;zMov&gt;</td>
<td>REAL</td>
<td>Additive offset values depending on the reference system:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Workpiece-related protection zone: Machine zero</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tool-related protection zone: Toolholder reference point</td>
</tr>
</tbody>
</table>

#### Note

A protection zone is only taken into account after the referencing of all geometry axes of the channel in which it has been activated.
Activation via NC/PLC interface signals

Only protection zones that have been preactivated via the part program (see paragraph below "Preactivation via part program") can be activated in the PLC user program via the NC/PLC interface signals:

- DB21, ... DBX8.0 - 9.1 = 1 (activate machine-related protection zone 1 - 10)
- DB21, ... DBX10.0 - 11.1 = 1 (activate channel-specific protection zone 1 - 10)

For preactivation, see paragraph below "Preactivation via part program"

The activation of preactivated protection zones must be performed prior to the traversing motion of the geometry axes. If the activation is performed during the traversing motion, these protection zones are not taken into account in the current traversing motion. Reaction:

- Alarm "10704 Protection zone monitoring is not guaranteed"
- DB21, ... DBX39.0 = 1 (protection zone monitoring not guaranteed)

Note

The activation of preactivated protection zones must be performed prior to the traversing motion of the geometry axes.

Automatic activation after the NC powers up

Protection zones that are to take effect immediately after the NC powers up can be specified via the following system variables:

- Channel-specific protection zones:
  $SC_PA_ACTIV_IMMED[<protection zone number>]
- Machine-specific protection zones:
  $SN_PA_ACTIV_IMMED[<protection zone number>]

Note

With automatic activation, no relative offset of a protection zone is possible.

Note

A protection zone is only taken into account after the referencing of all geometry axes of the channel in which it has been activated.

Preactivation via part program

Protection zones that are to be activated at a later time from the PLC user program must be preactivated in the part program:

CPROT or NPROT (<protection zone number>, 1)
Preactivated protection zones are displayed via the following NC/PLC interface signals:

- DB21, ... DB272.0 - 273.1 == 1 (machine-related protection zone 1 - 10 preactivated)
- DB21, ... DB274.0 - 275.1 == 1 (channel-specific protection zone 1 - 10 preactivated)

![Preactivated protection zone](image)

Figure 3-13  Example: Turning machine with preactivated protection zone for a sensor.

**Preactivation with conditional stop**

<table>
<thead>
<tr>
<th>NOTICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protection zone violation possible</td>
</tr>
</tbody>
</table>

If a preactivated protection zone with conditional stop is not activated in good time, the NC may no longer be able to stop before the protection zone in good time because the braking distance after the activation point has not been taken into account.

The preactivation of protection zones with conditional stop is performed in the part program via:

CPROT or NPROT (<protection zone number>, 3)

In the case of a preactivated protection zone with conditional stop, a traversing motion is **not** stopped before this if the traversing motion goes into the protection zone. A stop is only performed when the protection zone has been activated. This behavior is to enable uninterrupted machining controlled by the user when the protection zone is only required temporarily.

**Deactivation via part program**

A protection zone that has been activated via the part program or an NC/PLC interface signal can be deactivated again at any time from the part program with status = 0:

- Channel-specific protection zone:
  
  CPROT(n, 0)

- Machine-specific protection zone:
  
  NPROT(n, 0)
Deactivation via NC/PLC interface signal

Only protection zones that have been preactivated via a part program and activated via the NC/PLC interface signals, can be deactivated again via the NC/PLC interface signals:

- DB21, ... DBX8.0 to DBX9.1 = 0 (activate machine-related protection zone 1 - 10)
- DB21, ... DBX10.0 to DBX11.1 = 0 (activate channel-specific protection zone 1 - 10)

Protection zones that have been activated directly via a part program cannot be deactivated from the PLC user program.

Note

It is recommended that protection zones which are to be activated via the PLC user program, be configured specially for this. Preactivation in the part program is only useful for these protection zones.

Automatic deactivation via machine data parameterization

When executing the Geometry axis change and Transformation change functions, the active protection zones can be deactivated automatically. The setting is performed NC-specifically via the machine data:

MD10618 $MN_PROTAREA_GEOAX_CHANGE_MODE

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>The active protection zones are deactivated during the transformation change.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>The active protection zones remain active during the transformation change.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>The active protection zones are deactivated during the geometry axis change.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>The active protection zones remain active during the geometry axis change.</td>
</tr>
</tbody>
</table>

Display protection zone violation

Violations of activated protection zones or possible violations of preactivated protection zones, if they would be activated, are displayed via the following NC/PLC interface signals:

- DB21, ... DBX276.0 - 277.1 == 1 (machine-related protection zone 1 - 10 violated)
- DB21, ... DBX278.0 - 279.1 == 1 (channel-specific protection zone 1 - 10 violated)

Behavior in special system states

Block search with calculation

For block search with calculation, the last programmed activation state of a protection zone is always taken into account.

Program test

In the AUTOMATIC and MDI modes, activated and preactivated protection zones are also monitored in the "Program test" state.
NC RESET and end of program
The activation status of a protection zone is retained even after an NC RESET and end of program.

Memory requirements
The memory required for protection zones is parameterized via the following machine data:

- Persistent memory
  - MD18190 $MN_MM_NUM_PROTECT_AREA_NCK (number of available machine-defined protection zones)
  - MD28200 $MC_MM_NUM_PROTECT_AREA_CHAN (number of available channel-defined protection zones)

- Dynamic memory
  - MD28210 $MC_MM_NUM_PROTECT_AREA_ACTIVE (maximum number of protection zones that can be activated simultaneously in the channel)
  - MD28212 $MC_MM_NUM_PROTECT_AREA_CONTUR (maximum number of definable contour elements per protection zone)

See also
Definition via part program instruction (Page 135)
Definition as per system variable (Page 138)

3.3.6 Protection zone violation and temporary enabling of individual protection zones

Temporary enabling of protection zones
If a protection zone violation occurs when starting or during a traversing motion, under certain circumstances, the protection zone can be enabled, i.e. for temporary traversing. In AUTOMATIC and MDA mode as well as in JOG mode, the temporary enabling of protection zones is performed via operator actions.

A temporary enable is only possible for workpiece-related protection zones.

Tool-related protection zones must either be deactivated in the part program or via the NC/PLC interface in the "Preactivated" state.

Terminating the temporary enabling
Temporary enabling of a protection zone is terminated after the following events:

- NC RESET
- AUTOMATIC or MDA mode: The end of the block is outside the protection zone
- JOG mode: The end of the traversing motion is outside the protection zone
- The protection zone is activated
Behavior in the AUTOMATIC and MDA modes

In AUTOMATIC and MDA mode, no traversing motion is enabled into or through active protection zones:

- A traversing motion that would lead from outside into an active protection zone, is stopped at the end of the last block located outside the protection zone.
- A traversing motion that begins within an active protection zone is not started.

Temporary enabling of protection zones

If a traversing motion is stopped in AUTOMATIC or MDA mode because of a protection zone violation, this is indicated to the operator by an alarm. If the operator decides that the traversing motion can be continued, the crossing of protection zones can be enabled.

The enabling is only temporarily and is performed by triggering NC START:

\[ DB21, \ldots \text{DBX7.1} = 1 \] (NC START)

An alarm is displayed for each violated protection zone. An NC START signal must be triggered by the operator for each protection zone to be enabled.

The traversing motion is continued when all protection zones that have resulted in the stopping of the traversing motion, have been enabled.

Continuation of a traversing motion without temporary enabling

A traversing motion was stopped with an alarm because of a protection zone violation. If the relevant protection zone is set to the "Preactivated" state via the NC/PLC interface, the traversing motion can be continued with NC START, without the protection zone being temporarily enabled.

Increased protection against the enabling of protection zones

If the enabling of a protection zone is to be protected better than just by NC START, this must be performed by the machine manufacturer or user in the PLC user program.

Behavior in JOG mode

Simultaneous traversing of several geometry axes

In JOG mode, traversing motions can be performed simultaneously in several geometry axes. The traversing range of every participating geometry axis is limited axis-specifically at the start of the traversing motion with regard to the traversing range limits (software limit switches, working area limitation, etc.) and active protection zones. However, safe monitoring of all active protection zones cannot be guaranteed. The following is provided as feedback to the user:

- Alarm: "10704 Protection zone monitoring is not guaranteed"
- \[ \text{DB31, \ldots DBX39.0} = 1 \] (protection zone monitoring not guaranteed)
After the end of the traversing motion, the alarm is cleared automatically. If the current position is within an activated or preactivated protection zone, the following is performed:

- Alarm 10702 or 10703 is displayed
- Further traversing motions are disabled
- The NC/PLC interface signal for the relevant protection zone is set DB21, ... DBX276.0 - 277.1 or DBX278.0 - 279.1 (protection zone violated)

To continue, see the paragraph below "Temporary enabling of protection zones" and Section "Activating and deactivating protection zones (Page 140)".

**Example:** Three protection zones activated and simultaneous traversing of two geometry axes

![Diagram showing motion range of geometry axes](image-url)

Figure 3-14  Motion range of the geometry axes at the start time

At the start time of the traversing motions of the axes X and Y, the axial traversing range limits are determined from the start time:

- **X axis**
  - Positive traversing direction: Protection zone 2
  - Negative traversing direction: Absolute traversing range limit, e.g. software limit switch
- **Y axis**
  - Positive traversing direction: Protection zone 1
  - Negative traversing direction: Absolute traversing range limit, e.g. working area limitation
The resulting maximum traversing range at the start time does not take protection zone 3 into account. Therefore, a protection zone violation in protection zone 3 is possible.

**Note**
Activated and preactivated protection zones are also monitored in the manual operating modes JOG, INC and DRF.

**Limitation of traversing motion of an axis**
If the traversing motion of an axis is limited because it has reached a protection zone, then a selfresetting alarm "Protection zone reached in JOG" is generated. The alarm text specifies the violated protection zone and the relevant axis. It is assured that no protection zone will be violated when an axis is traversing in JOG. (This response is analogous to approaching software limit switches or a working area limitation.)

The alarm is reset:
- When an axis is traversed along a path that does not lead into the protection zone
- When the protection zone is enabled
- On **NC RESET**

If an axis starts to move towards a protection zone when it is at a protection zone limit, then a selfresetting alarm "Protection zone reached in JOG" is output and the motion is not started.

**Temporary enabling of protection zones**
If a traversing motion is started within or on the limit of an activated protection zone, alarm 10702 or 10703 is displayed and the traversing motion is not started. The traversing motion can be performed when the relevant protection zone is temporarily enabled. The following actions must be performed for this:
- Create a positive edge at the NC/PLC interface signal:
  DB21, ... DBX1.1 (enable protection zone)
- Start the same traversing motion again

**Note**
The NC/PLC interface signal "Protection zone violated" is still set when the temporarily enabled protection zone is traversed:

DB21, ... DBX276.0 – 277.1 or DBX278.0 – 279.1

The enable is reset if a motion is started that does not lead into the enabled protection zone.

If further protection zones are affected by the traversing motion, additional alarms 10702 or 10703 are displayed for each protection zone. The protection zones displayed in the alarms can be enabled by creating further positive edges on the NC/PLC interface signal:

DB21, ... DBX1.1 (enable protection zone)

**Behavior at change of operating mode**
The protection zones temporarily enabled in JOG mode are retained after a change to AUTOMATIC or MDA mode. The temporary enables set the in the AUTOMATIC or MDA mode are also retained after a change to JOG mode.
3.3 Protection zones

Reset of an enable

The enable is reset internally and on the NC/PLC interface at the next standstill of a geometry axis for which the temporarily enabled protection zone has been completely exited:

DB21, … DBX276.0 - 277.1 or DBX278.0 - 279.1 = 0 (protection zone violated)

Behavior with axis replacement

With regard to the protection zones after an axis replacement, the starting position is the last position approached in the relinquishing channel. Traversing motions in the channel taking over are not taken into account. For this reason, you must make sure that an axis replacement is not performed from a position with protection zone violation.

Behavior for superimposed motions

Superimposed motions that are included in the main run, cannot be taken into account by the block preparation with regard to the active protection zones.

This results in the following reactions:

- Alarm: "10704 Protection zone monitoring is not guaranteed"
- DB31, … DBX39.0 = 1 (protection zone monitoring not guaranteed)

3.3.7 Restrictions in protection zones

Restrictions in protection-zone monitoring

No protection-zone monitoring is possible under the following conditions:

- Orientation axes
- Protection-zone monitoring for fixed machine-related protection zones with TRANSMIT or peripheral surface transformation. Exception: Protection zones defined with rotation symmetry around the spindle axis. Here, no DRF offset must be active.
- Mutual monitoring of tool-related protection zones

Positioning axes

For positioning axes, only the programmed block end point is monitored. An alarm is displayed during the traversing motion of the positioning axes:

Alarm: "10704 Protection-zone monitoring is not guaranteed".

Axis interchange

If an axis intended for the axis interchange is not active in a channel, the position of the axis last approached in the channel is taken as the current position. If this axis has not yet been traversed in the channel, zero is taken as the position.
Machine-related protection zones

A machine-related protection zone or its contour is defined using the geometry axis, i.e. with reference to the basic coordinate system (BCS) of a channel. In order that correct protection-zone monitoring can take place in all channels in which the machine-related protection zone is active, the basic coordinate system (BCS) of all affected channels must be identical (position of the coordinate point of origin with respect to the machine zero point and orientation of the coordinate axes).

3.3.8 Checking for protection zone violation, working area limitation and software limit switches (CALCPOSI)

Function

As of the start position, the CALCPOSI() function checks whether active limits have been violated along the traversing distance in the workpiece coordinate system (WCS) with regard to the geometry axes.

If the distance cannot be fully traversed because of limits, a positive, decimal-coded status value and the maximum possible traversing distance are returned.

Definition

```
```

Syntax

```
<Status> = CALCPOSI(VAR <Start>, VAR <Dist>, VAR <Limit>, VAR <MaxDist>, <System>, <TestLim>)
```

Meaning

```
<table>
<thead>
<tr>
<th>CALCPOSI:</th>
<th>Test for limit violations with regard to the geometry axes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessing stop:</td>
<td>No</td>
</tr>
<tr>
<td>Alone in the block:</td>
<td>Yes</td>
</tr>
</tbody>
</table>
```
### Function return value. Negative values indicate error states.

**Data type:** INT  
**Range of values:** \(-8 \leq x \leq 100000\)

<table>
<thead>
<tr>
<th>Values</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The distance can be traversed completely.</td>
</tr>
<tr>
<td>-1</td>
<td>At least one component is negative in &lt;Limit&gt;.</td>
</tr>
<tr>
<td>-2</td>
<td>Error in a transformation calculation. Example: The traversing distance passes through a singularity so that the axis positions cannot be defined.</td>
</tr>
<tr>
<td>-3</td>
<td>The specified traversing distance &lt;Dist&gt; and the maximum possible traversing distance &lt;MaxDist&gt; are linearly dependent. Note: Can only occur in conjunction with &lt;TestLim&gt;, bit 4 == 1.</td>
</tr>
<tr>
<td>-4</td>
<td>The projection of the traversing direction contained in &lt;Dist&gt; on to the limitation surface is the zero vector, or the traversing direction is perpendicular to the violated limitation surface. Note: Can only occur in conjunction with &lt;TestLim&gt;, bit 5 == 1.</td>
</tr>
<tr>
<td>-5</td>
<td>In &lt;TestLim&gt;, bit 4 == 1 AND bit 5 == 1</td>
</tr>
<tr>
<td>-6</td>
<td>At least one machine axis that has to be considered for checking the traversing limits has not been homed.</td>
</tr>
<tr>
<td>-7</td>
<td>Collision avoidance function: Invalid definition of the kinematic chain or the protection zones.</td>
</tr>
<tr>
<td>-8</td>
<td>Collision avoidance function: This command cannot be executed because of insufficient memory.</td>
</tr>
</tbody>
</table>

### Units digit

- **1**: Software limit switches are limiting the traversing distance  
- **2**: Working area limits are limiting the traversing distance  
- **3**: Protection zones are limiting the traversing distance

### Tens digit

- **1x**: The initial value violates the limit  
- **2x**: The specified straight line violates the limit. This value is also returned if the end point does not violate any limit itself, but the path from the starting point to the end point would cause a limit value to be violated (e.g. by passing through a protection zone, curved software limit switches in the WCS for non-linear transformations, e.g. transmit).
### 3.3 Protection zones

#### <Status>:

##### (Part 3)

<table>
<thead>
<tr>
<th>Hundreds digit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1xx AND units digit == 1 or 2:</td>
<td>The positive limit value has been violated.</td>
</tr>
<tr>
<td>AND units digit == 3 (^\d):</td>
<td>An NC-specific protection zone has been violated.</td>
</tr>
<tr>
<td>2xx AND units digit == 1 or 2:</td>
<td>The negative limit value has been violated.</td>
</tr>
<tr>
<td>AND units digit == 3 (^\d):</td>
<td>A channel-specific protection zone has been violated.</td>
</tr>
</tbody>
</table>

##### (Part 4)

<table>
<thead>
<tr>
<th>Thousands digit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1xxx AND units digit == 1 or 2:</td>
<td>Factor with which the axis number is multiplied that violates the limit. Numbering of the axes begins with 1.</td>
</tr>
<tr>
<td>Reference:</td>
<td></td>
</tr>
<tr>
<td>● Software limit switch: Machine axes</td>
<td></td>
</tr>
<tr>
<td>● Working area limitation: Geometry axes</td>
<td></td>
</tr>
<tr>
<td>AND units digit == 3 (^\d):</td>
<td>Factor with which the number of the violated protection zone is multiplied.</td>
</tr>
</tbody>
</table>

##### (Part 5)

<table>
<thead>
<tr>
<th>Hundred thousands digit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xxxxx Hundred thousands digit == 0:</td>
<td>&lt;Dist&gt; remains unchanged</td>
</tr>
<tr>
<td>1xxxxx A direction vector is returned in &lt;Dist&gt; which defines the further motion direction on the limitation surface.</td>
<td></td>
</tr>
<tr>
<td>Can only occur with the following supplementary conditions:</td>
<td></td>
</tr>
<tr>
<td>● Software limit switch or working area limit violated (not in the starting point)</td>
<td></td>
</tr>
<tr>
<td>● A transformation is not active</td>
<td></td>
</tr>
<tr>
<td>● &lt;TestID&gt;, bit 4 or bit 5 == 1</td>
<td></td>
</tr>
</tbody>
</table>

#### <Start>:

Reference to a vector with the start positions:
- <Start> [0]: Abscissa
- <Start> [1]: Ordinate
- <Start> [2]: Applicate

Parameter type: Input
Data type: VAR REAL [3]
Range of values: -max. REAL value ≤ x[n] ≤ +max. REAL value
A3: Axis Monitoring, Protection Zones

3.3 Protection zones

|<Dist>| Reference to a vector.  
**Input:** Incremental traversing distance  
- <Dist> [0]: Abscissa  
- <Dist> [1]: Ordinate  
- <Dist> [2]: Applicate  

**Output (only for set hundred thousands digit in <Status>):**  
<Dist> contains a unit vector \( \mathbf{v} \) as output value which defines the further traversing direction in the WCS.  

**Case 1:** Formation of vector \( \mathbf{v} \) for <TestID>, bit 4 == 1  
The input vectors <Dist> and <MaxDist> span the motion plane. This plane is cut by the violated limitation surface. The intersecting line of the two planes defines the direction of vector \( \mathbf{v} \). The orientation (sign) is selected so that the angle between the input vector <MaxDist> and \( \mathbf{v} \) is not greater than 90 degrees.  

**Case 2:** Formation of vector \( \mathbf{v} \) for <TestID>, bit 5 == 1  
Vector \( \mathbf{v} \) is the unit vector in the projection direction of the traversing vector contained in <Dist> on the limitation surface. If the projection of the traversing vector on the limitation surface is the zero vector, an error is returned.  

**Parameter type:** Input/output  
**Data type:** VAR REAL [3]  
**Range of values:** -max. REAL value ≤ x[n] ≤ +max. REAL value

|<Limit>| Reference to an array of length 5.  
- <Limit> [0 - 2]: Minimum clearance of the geometry axes to the limits:  
  - <Limit> [0]: Abscissa  
  - <Limit> [1]: Ordinate  
  - <Limit> [2]: Applicate  
  The minimum clearances are observed with:  
  - Working area limitation: No restrictions  
  - Software limit switch: If no transformation is active, or a transformation is active in which a clear assignment of the geometry axes to the linear machine axes is possible, e.g. 5-axis transformations.  
- <Limit> [3]: Contains the minimum clearance for linear machine axes which, for example, cannot be assigned a geometry axis because of a non-linear transformation. This value is also used as limit value for the monitoring of the conventional protection zones and the collision avoidance protection zones.  
- <Limit> [4]: Contains the minimum clearance for rotary machine axes which, for example, cannot be assigned a geometry axis because of a non-linear transformation.  

**Note**  
This value is only active for the monitoring of the software limit switches for special transformations.  

**Parameter type:** Input  
**Data type:** VAR REAL [5]  
**Range of values:** -max. REAL value ≤ x[n] ≤ +max. REAL value
Reference to a vector with the incremental traversing distance in which the specified minimum clearance of an axis limit is not violated by any of the relevant machine axes:

- \(<\text{Dist}>[0]\): Abscissa
- \(<\text{Dist}>[1]\): Ordinate
- \(<\text{Dist}>[2]\): Applicate

If the traversing distance is not restricted, the contents of this return parameter are the same as the contents of \(<\text{Dist}>\).

With \(<\text{TestID}>\), bit 4 == 1: \(<\text{Dist}>\) and \(<\text{MaxDist}>\)

\(<\text{MaxDist}>\) and \(<\text{Dist}>\) must contain vectors as input values that span a motion plane. The two vectors must be mutually linearly dependent. The absolute value of \(<\text{MaxDist}>\) is arbitrary. For the calculation of the motion direction, see the description for \(<\text{Dist}>\).

Parameter type: Output
Data type: VAR REAL [3]
Range of values: -max. REAL value ≤ x[n] ≤ +max. REAL value

Measuring system (inch/metric) for position and distance specifications (optional)

Parameter type: Output
Data type: BOOL
Default value: FALSE

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>Measuring system corresponding to the currently active G function from G group 13 (G70, G71, G700, G710). <strong>Note</strong> If G70 is active and the basic system is metric (or G71 is active and the basic system is inch), the system variables $AA_IW and $AA_MW are provided in the basic system and, if used, must be converted for (\text{CALCPOSI()}).</td>
</tr>
<tr>
<td>TRUE</td>
<td>Measuring system corresponding to the set basic system: MD52806 $MN_ISO_SCALING_SYSTEM</td>
</tr>
</tbody>
</table>

Bit-coded selection of the limits to be monitored (optional)

Parameter type: Output
Data type: INT
Default value: Bit 0, 1, 2, 3 == 1 (15)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Decimal</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>Software limit switch</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>Working area limitation</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Activated conventional protection zones</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>Preactivated conventional protection zones</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>With violated software limit switches or working area limits in (&lt;\text{Dist}&gt;), return the traversing direction as in <strong>Case 1</strong> (see above).</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>With violated software limit switches or working area limits in (&lt;\text{Dist}&gt;), return the traversing direction as in <strong>Case 2</strong> (see above).</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>Activated collision avoidance protection zones</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>Preactivated collision avoidance protection zones</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>Pairs of activated and preactivated collision avoidance protection zones</td>
</tr>
</tbody>
</table>

1) If several protection zones are violated, the protection zone with the greatest restriction on the specified traversing distance is returned.
Example

In the example (see figure), the active software limit switches and working area limits in the X-Y plane and the following three protection zones are displayed:

- **C2**: Tool-related, channel-specific protection zone, active, circular, radius = 2 mm
- **C4**: Workpiece-related, channel-specific protection zone, preactivated, square, side length = 10 mm
- **N3**: NC-specific protection zone, active, rectangular, side length = 10 mm x 15 mm

The protection zones and working area limits are defined first in the NC program. The `CALCPOSI()` function is then called with different parameterizations.

---

**NC program**

**Program code**

```
N10 DEF REAL _STARTPOS[3]
N20 DEF REAL _MOVDIST[3]
N30 DEF REAL _DLIMIT[5]
N40 DEF REAL _MAXDIST[3]
N50 DEF INT _SB
N60 DEF INT _STATUS
N70 CPRTDEF(2, TRUE, 0)
N80 G17 G1 X-2 Y0
N90 G3 I2 X2
N100 I-2 X-2
N110 EXECUTE(_SB)
```
Program code

; Workpiece-related protection zone C4
N120 CPROTDEF(4, FALSE, 0)
N130 G17 G1 X0 Y15
N140 X10
N150 Y25
N160 X0
N170 Y15
N180 EXECUTE(_SB)

; Machine-related protection zone N3
N190 NPROTDEF(3, FALSE, 0)
N200 G17 G1 X10 Y5
N210 X25
N220 Y15
N230 X10
N240 Y5
N250 EXECUTE(_SB)

; Activate or preactivate protection zones
N260 CPROT(2, 2, 0, 0, 0)
N270 CPROT(4, 1, 0, 0, 0)
N280 NPROT(3, 2, 0, 0, 0)

; Define working area limits
N290 G25 XX=-10 YY=-10
N300 G26 XX=20 YY=21
N310 _STARTPOS[0] = 0.
N320 _STARTPOS[1] = 0.
N330 _STARTPOS[2] = 0.
N340 _MOVDIST[0] = 35.
N360 _MOVDIST[2] = 0.
N370 _DLIMIT[0] = 0.
N380 _DLIMIT[1] = 0.
N390 _DLIMIT[2] = 0.
N400 _DLIMIT[3] = 0.
N410 _DLIMIT[4] = 0.
N420 _STATUS = CALCPOSI(_STARTPOS, _MOVDIST, _DLIMIT, _MAXDIST)
N430 _STATUS = CALCPOSI(_STARTPOS, _MOVDIST, _DLIMIT, _MAXDIST,,3)
N440 _STATUS = CALCPOSI(_STARTPOS, _MOVDIST, _DLIMIT, _MAXDIST,,1)
N450 _STARTPOS[0] = 5.
N470 _STARTPOS[2] = 0.
N480 _MOVDIST[0] = 0.
N500 _MOVDIST[2] = 0.
N510 _STATUS = CALCPOSI(_STARTPOS, _MOVDIST, _DLIMIT, _MAXDIST,,14)
N520 _STATUS = CALCPOSI(_STARTPOS, _MOVDIST, _DLIMIT, _MAXDIST,, 6)
N530 _DLIMIT[1] = 2.
N540 _STATUS = CALCPOSI(_STARTPOS, _MOVDIST, _DLIMIT, _MAXDIST,, 6)
N550 _STARTPOS[0] = 27.
N560 _STARTPOS[1] = 17.1
N570 _STARTPOS[2] = 0.
Program code

N580 _MOVDIST[0] = -27.
N590 _MOVDIST[1] = 0.
N600 _MOVDIST[2] = 0.
N620 _STATUS = CALCPOSI(_STARTPOS, _MOVDIST, _DLIMIT, _MAXDIST,,12)
N630 _STARTPOS[0] = 0.
N640 _STARTPOS[1] = 0.
N650 _STARTPOS[2] = 0.
N660 _MOVDIST[0] = 0.
N690 TRANS X10
N700 AROT Z45
N710 _STATUS = CALCPOSI(_STARTPOS, _MOVDIST, _DLIMIT, _MAXDIST)
; Delete frames from N690 and N700 again
N720 TRANS
N730 _STARTPOS[0] = 0.
N750 _STARTPOS[2] = 0.
; Vectors _MOVDIST and _MAXDIST define the motion plane
N760 _MOVDIST[0] = 30.
N780 _MOVDIST[2] = 0.
N790 _MAXDIST[0] = 1.
N800 _MAXDIST[1] = 0.
N820 _STATUS = CALCPOSI(_STARTPOS, _MOVDIST, _DLIMIT, _MAXDIST,,17)
N830 M30

Results of CALCPOSI():

<table>
<thead>
<tr>
<th>N</th>
<th>&lt;status&gt;</th>
<th>&lt;MaxDist&gt;[0] X</th>
<th>&lt;MaxDist&gt;[1] Y</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>420</td>
<td>3123</td>
<td>8.040</td>
<td>4.594</td>
<td>N3 is violated.</td>
</tr>
<tr>
<td>430</td>
<td>1122</td>
<td>20.000</td>
<td>11.429</td>
<td>No limit is violated.</td>
</tr>
<tr>
<td>440</td>
<td>1121</td>
<td>30.000</td>
<td>17.143</td>
<td>Only monitoring of the software limit switches still active.</td>
</tr>
<tr>
<td>510</td>
<td>4213</td>
<td>0.000</td>
<td>0.000</td>
<td>Starting point violates C4</td>
</tr>
<tr>
<td>520</td>
<td>0000</td>
<td>0.000</td>
<td>-27.000</td>
<td>Preactivated C4 is not monitored. The specified distance can be traversed completely.</td>
</tr>
<tr>
<td>540</td>
<td>2222</td>
<td>0.000</td>
<td>-25.000</td>
<td>Because _DLIMIT[1] = 2, the traversing distance is restricted by the working area limitation.</td>
</tr>
<tr>
<td>620</td>
<td>4223</td>
<td>-13.000</td>
<td>0.000</td>
<td>Clearance to C4 is a total of 4 mm due to C2 and _DLIMIT[3]. Clearance C2 – N3 of 0.1 mm does not result in limitation of the traversing distance.</td>
</tr>
</tbody>
</table>
Supplementary conditions

**Axis status**
All machine axes considered by CALCPOSI() must be homed.

**Circle-related distance specifications**
All circle-related distance specifications are always interpreted as radius specifications. This must be taken into account particularly for transverse axes with activated diameter programming (DIAMON/DIAM90).

**Traversing distance reduction**
If the specified traversing distance of an axis is limited, the traversing distance of the other axes is also reduced proportionally in the <MaxDist> return value. The resulting end point is therefore still on the specified path.

**Rotary axes**
Rotary axes are only monitored when they are not modulo rotary axes.

It is permissible that no software limit switches, working area limits or protection zones are defined for one or more of the relevant axes.

**Software limit switch and working area limitation status**
Software limit switches and working area limits are only taken into account if they are active during the execution of CALCPOSI(). The status can be influenced, for example, via:
- Machine data: MD21020 $MC_WORKAREA_WITH_TOOL_RADIUS
- Setting data: $AC_WORKAREA_CS...
- NC/PLC interface signals DB31, ... DBX12.2 / 3
- Commands: GWALIMON/WALIMOF

**Software limit switches and transformations**
With CALCPOSI(), the positions of the machine axes (MCS) cannot always be clearly determined from the positions of the geometry axes (WCS) during various kinematic transformations (e.g. TRANSMIT) because of ambiguities at certain positions of the traversing distance. In normal traversing operation, the uniqueness generally results from the history and the condition that a continuous motion in the WCS must correspond to a continuous motion in...
the MCS. Therefore, when monitoring the software limit switches, the machine position at the
time when \texttt{CALCPOSI()} is executed is used to resolve the ambiguity in such cases.

\textbf{Note}

\textbf{Preprocessing stop}

When using \texttt{CALCPOSI()} in conjunction with transformations, it is the sole responsibility of
the user to program a preprocessing stop (\texttt{STOPRE}) with the preprocessing before
\texttt{CALCPOSI()} for the synchronization of the machine axis positions.

\textbf{Protection zone clearance and conventional protection zones}

With conventional protection zones, there is no guarantee that the safety clearance set in
\texttt{<Limit>[3]} is maintained for all protection zones during a traversing movement on the
specified path. It is only guaranteed that no protection zone will be violated when the end point
returned in \texttt{<Dist>} is extended by the safety clearance in the traversing direction. However,
the straight line can pass very close to a protection zone.

\textbf{Protection zone clearance and collision avoidance protection zones}

With collision avoidance protection zones, there is a guarantee that the safety clearance set
in \texttt{<Limit>[3]} is maintained for all protection zones during a traversing movement on the
specified path.

The safety clearance specified in \texttt{<Limit>[3]} only takes effect when the following applies:

\texttt{<Limit>[3]} > (MD10619 $MN_{\text{COLLISION\_TOLERANCE}})

If bit 4 is set in \texttt{<TestLim>} (see above) (value 16) (calculation of the further direction), then
the direction vector received in _MOVDIST is only valid when the hundred thousands digit is
set in the function return value (status). If such a direction cannot be determined because either
protection zones have been violated or a transformation is active, the input value in _MOVDIST
remains unchanged, no further error message is issued.

\textbf{References}

- Further information on working area limitation can be found in:
  Programming Manual, Fundamentals
  Section "Supplementary commands" > "Working area limitation"
- Further information on software limit switches can be found in:
  Function Manual, Basic Functions; (A3) Axis monitoring, protection zones,
  Section "Axis monitoring" > "Limit switch monitoring"
3.4 Supplementary conditions

3.4.1 Axis monitoring functions

Settings

For correct operation of the monitoring, the following settings must be made or checked, in addition to the machine data mentioned:

**General**

- MD31030 $MA_LEADSCREW_PITCH (leadscrew pitch)
- MD31050 $MA_DRIVE_AX_RATIO_DENOM (denominator load gearbox)
- MD31060 $MA_DRIVE_AX_RATIO_NUMERA (numerator load gearbox)
- MD31070 $MA_DRIVE_ENC_RATIO_DENOM (denominator measuring gearbox)
- MD31080 $MA_DRIVE_ENC_RATIO_NUMERA (numerator measuring gearbox)
- MD32810 $MA_EQUIV_SPEEDCTRL_TIME (equivalent time constant speed control loop for feedforward control)

**Encoder resolution**

For the appropriate machine data, see Section "G2: Velocities, setpoint / actual value systems, closed-loop control (Page 317)".

**Only drives with analog speed setpoint interface**

- MD32260 $MA_RATED_VELO (rated motor speed)
- MD32250 $MA_RATED_OUTVAL (rated output voltage)

3.5 Examples

3.5.1 Axis monitoring functions

3.5.1.1 Working area limitation in WCS/SZS

**Available channel axes**

Four axes are defined in the channel: X, Y, Z and A

The A-axis is a rotary axis (not modulo).

**Parameterize the number of working area limitation groups**

Three working area limitation groups will be provided:
MD28600 $MC_MM_NUM_WORKAREA_CS_GROUP = 3

Define the working area limitation groups

Additionally two working area limitation groups will be defined:

**Working area limitation group 1**

In the first working area limitation group the axes in the SZS coordinate system will be limited:

- X axis in the plus direction: 10 mm
- X axis in the minus direction: No limitation
- Y axis in the plus direction: No limitation
- Y axis in the minus direction: 25 mm
- Z axis in the plus direction: No limitation
- Z axis in the minus direction: No limitation
- A axis in the plus direction: 10 degrees
- A axis in the minus direction: -40 degrees

The system variables are assigned as follows:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1 $P_WORKAREA_CS_COORD_SYSTEM[1]=3</td>
<td>; The working area limitation of the working area limitation group 1 applies in the SZS.</td>
</tr>
<tr>
<td>N10 $P_WORKAREA_CS_PLUS_ENABLE[1,X]=TRUE</td>
<td></td>
</tr>
<tr>
<td>N11 $P_WORKAREA_CS_LIMIT_PLUS[1,X]=10</td>
<td></td>
</tr>
<tr>
<td>N12 $P_WORKAREA_CS_MINUS_ENABLE[1,X]=FALSE</td>
<td></td>
</tr>
<tr>
<td>N20 $P_WORKAREA_CS_PLUS_ENABLE[1,Y]=FALSE</td>
<td></td>
</tr>
<tr>
<td>N22 $P_WORKAREA_CS_MINUS_ENABLE[1,Y]=TRUE</td>
<td></td>
</tr>
<tr>
<td>N23 $P_WORKAREA_CS_LIMIT_MINUS[1,Y]=25</td>
<td></td>
</tr>
<tr>
<td>N30 $P_WORKAREA_CS_PLUS_ENABLE[1,Z]=FALSE</td>
<td></td>
</tr>
<tr>
<td>N32 $P_WORKAREA_CS_MINUS_ENABLE[1,Z]=FALSE</td>
<td></td>
</tr>
<tr>
<td>N40 $P_WORKAREA_CS_PLUS_ENABLE[1,A]=TRUE</td>
<td></td>
</tr>
<tr>
<td>N41 $P_WORKAREA_CS_LIMIT_PLUS[1,A]=10</td>
<td></td>
</tr>
<tr>
<td>N42 $P_WORKAREA_CS_MINUS_ENABLE[1,A]=TRUE</td>
<td></td>
</tr>
<tr>
<td>N43 $P_WORKAREA_CS_LIMIT_MINUS[1,A]=40</td>
<td></td>
</tr>
</tbody>
</table>

**Working area limitation group 2**

In the second working area limitation group the axes in the WCS coordinate system can be limited:

- X axis in the plus direction: 10 mm
- X axis in the minus direction: No limitation
- Y axis in the plus direction: 34 mm
- Y axis in the minus direction: -25 mm
• Z axis in the plus direction: No limitation
• Z axis in the minus direction: -600 mm
• A axis in the plus direction: No limitation
• A axis in the minus direction: No limitation

The system variables are assigned as follows:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N51 $P__WORKAREA__CS_COORD_SYSTEM[2]=1</td>
<td>; The working area limitation of working area limitation group 2 applies in the WCS.</td>
</tr>
<tr>
<td>N60 $P__WORKAREA__CS_PLUS_ENABLE[2,X]=TRUE</td>
<td></td>
</tr>
<tr>
<td>N61 $P__WORKAREA__CS_LIMIT_PLUS[2,X]=10</td>
<td></td>
</tr>
<tr>
<td>N62 $P__WORKAREA__CS_MINUS_ENABLE[2,X]=FALSE</td>
<td></td>
</tr>
<tr>
<td>N70 $P__WORKAREA__CS_PLUS_ENABLE[2,Y]=TRUE</td>
<td></td>
</tr>
<tr>
<td>N73 $P__WORKAREA__CS_LIMIT_PLUS[2,Y]=34</td>
<td></td>
</tr>
<tr>
<td>N72 $P__WORKAREA__CS_MINUS_ENABLE[2,Y]=TRUE</td>
<td></td>
</tr>
<tr>
<td>N73 $P__WORKAREA__CS_LIMIT_MINUS[2,Y]=–25</td>
<td></td>
</tr>
<tr>
<td>N80 $P__WORKAREA__CS_PLUS_ENABLE[2,Z]=FALSE</td>
<td></td>
</tr>
<tr>
<td>N82 $P__WORKAREA__CS_MINUS_ENABLE[2,Z]=TRUE</td>
<td></td>
</tr>
<tr>
<td>N83 $P__WORKAREA__CS_LIMIT_PLUS[2,Z]=–600</td>
<td></td>
</tr>
<tr>
<td>N90 $P__WORKAREA__CS_PLUS_ENABLE[2,A]=FALSE</td>
<td></td>
</tr>
<tr>
<td>N92 $P__WORKAREA__CS_MINUS_ENABLE[2,A]=FALSE</td>
<td></td>
</tr>
</tbody>
</table>

Activate working area limitation group 2

In order to activate the working area limitation group 2, the following instruction must exist in the part program:

```
... 
N100 WALCS2 ... 
... 
```
### 3.5.2 Protection zones

#### 3.5.2.1 Definition and activation of protection zones

**Requirement**

The following internal protection zones are to be defined for a turning machine:

- One machine- and workpiece-related protection zone for the spindle chuck, without limitation in the third dimension
- One channel-specific protection zone for the workpiece, without limitation in the third dimension
- One channel-specific, tool-related protection zone for the toolholder, without limitation in the third dimension

The workpiece zero is placed on the machine zero to define the protection zone for the workpiece.

When activated, the protection zone is then offset by 100 mm in the Z axis in the positive direction.

![Example of protection zones on a turning machine](image)

**Figure 3-15** Example of protection zones on a turning machine
Protection zone definition in the part program

Table 3-1 Part program excerpt for protection zone definition:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEF INT AB</td>
<td></td>
</tr>
<tr>
<td>G18</td>
<td>Definition of the working plane</td>
</tr>
<tr>
<td>NPROTDEF(1,FALSE,0,0,0)</td>
<td>Start of definition: Protection zone for spindle chuck</td>
</tr>
<tr>
<td>G01 X100 Z0</td>
<td>Contour definition: 1. Contour element</td>
</tr>
<tr>
<td>G01 X-100 Z0</td>
<td>Contour definition: 2. Contour element</td>
</tr>
<tr>
<td>G01 X-100 Z110</td>
<td>Contour definition: 3. Contour element</td>
</tr>
<tr>
<td>G01 X100 Z110</td>
<td>Contour definition: 4. Contour element</td>
</tr>
<tr>
<td>G01 X100 Z0</td>
<td>Contour definition: 5. Contour element</td>
</tr>
<tr>
<td>EXECUTE(AB)</td>
<td>End of definition: Protection zone for spindle chuck</td>
</tr>
<tr>
<td>CPROTDEF(1,FALSE,0,0,0)</td>
<td>Start of definition: Protection zone for workpiece</td>
</tr>
<tr>
<td>G01 X80 Z0</td>
<td>Contour definition: 1. Contour element</td>
</tr>
<tr>
<td>G01 X-80 Z0</td>
<td>Contour definition: 2. Contour element</td>
</tr>
<tr>
<td>G01 X-80 Z40</td>
<td>Contour definition: 3. Contour element</td>
</tr>
<tr>
<td>G01 X80 Z40</td>
<td>Contour definition: 4. Contour element</td>
</tr>
<tr>
<td>G01 X80 Z0</td>
<td>Contour definition: 5. Contour element</td>
</tr>
<tr>
<td>EXECUTE(AB)</td>
<td>End of definition: Protection zone for workpiece</td>
</tr>
<tr>
<td>CPROTDEF(2,TRUE,0,0,0)</td>
<td>Start of definition: Protection zone for toolholder</td>
</tr>
<tr>
<td>G01 X0 Z-50</td>
<td>Contour definition: 1. Contour element</td>
</tr>
<tr>
<td>G01 X-190 Z-50</td>
<td>Contour definition: 2. Contour element</td>
</tr>
<tr>
<td>G03 X-210 Z-30 I-20</td>
<td>Contour definition: 3. Contour element</td>
</tr>
<tr>
<td>G01 X-210 Z20</td>
<td>Contour definition: 4. Contour element</td>
</tr>
<tr>
<td>G01 X0 Z50</td>
<td>Contour definition: 5. Contour element</td>
</tr>
<tr>
<td>G01 X0 Z-50</td>
<td>Contour definition: 6. Contour element</td>
</tr>
<tr>
<td>EXECUTE(AB)</td>
<td>End of definition: Protection zone for toolholder</td>
</tr>
</tbody>
</table>

Protection zone definition with system variables

Table 3-2 Protection zone: Spindle chuck

<table>
<thead>
<tr>
<th>System variable</th>
<th>Value</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SN_PA_ACTIV_IMMED[0]</td>
<td>0</td>
<td>; Protection zone for spindle chuck not immediately active</td>
</tr>
<tr>
<td>$SN_PA_T_W[0]</td>
<td>&quot; &quot;</td>
<td>; Machine-related protection zone for spindle chuck</td>
</tr>
<tr>
<td>$SN_PA_OR[0]</td>
<td>1</td>
<td>; Orientation of the protection zone: 1= 3, 1= 3rd and 1st geometry axis</td>
</tr>
<tr>
<td>$SN_PA_LIM_3DIM[0]</td>
<td>0</td>
<td>; Type of limitation in the third dimension: 0 = No limit</td>
</tr>
<tr>
<td>System variable</td>
<td>Value</td>
<td>Remark</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>$SN_PA_PLUS_LIM[0]$</td>
<td>0</td>
<td>Value of the limit in the positive direction in the 3rd dimension</td>
</tr>
<tr>
<td>$SN_PA_MINUS_LIM[0]$</td>
<td>0</td>
<td>Value of the limitation in the negative direction in the 3rd dimension</td>
</tr>
<tr>
<td>$SN_PA_CONT_NUM[0]$</td>
<td>4</td>
<td>Number of valid contour elements</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,0]$</td>
<td>1</td>
<td>Contour type[i] : 1 = G1 for even, Protection zone for spindle chuck, contour element 0</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,1]$</td>
<td>1</td>
<td>Contour type[i] : 1 = G1 for even, Protection zone for spindle chuck, contour element 1</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,2]$</td>
<td>1</td>
<td>Contour type[i] : 1 = G1 for even, Protection zone for spindle chuck, contour element 2</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,3]$</td>
<td>1</td>
<td>Contour type[i] : 1 = G1 for even, Protection zone for spindle chuck, contour element 3</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,4]$</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined, Protection zone for spindle chuck, contour element 4</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,5]$</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined, Protection zone for spindle chuck, contour element 5</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,6]$</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined, Protection zone for spindle chuck, contour element 6</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,7]$</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined, Protection zone for spindle chuck, contour element 7</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,8]$</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined, Protection zone for spindle chuck, contour element 8</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,9]$</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined, Protection zone for spindle chuck, contour element 9</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,0]$</td>
<td>-100</td>
<td>Endpoint of contour[i], ordinate value Protection zone for spindle chuck, contour element 0</td>
</tr>
<tr>
<td>$SN_PA_CONT ORD[0,1]$</td>
<td>-100</td>
<td>Endpoint of contour[i], ordinate value Protection zone for spindle chuck, contour element 1</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,2]$</td>
<td>100</td>
<td>Endpoint of contour[i], ordinate value Protection zone for spindle chuck, contour element 2</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,3]$</td>
<td>100</td>
<td>Endpoint of contour[i], ordinate value Protection zone for spindle chuck, contour element 3</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,4]$</td>
<td>0</td>
<td>Endpoint of contour[i], ordinate value Protection zone for spindle chuck, contour element 4</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,5]$</td>
<td>0</td>
<td>Endpoint of contour[i], ordinate value Protection zone for spindle chuck, contour element 5</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,6]$</td>
<td>0</td>
<td>Endpoint of contour[i], ordinate value Protection zone for spindle chuck, contour element 6</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,7]$</td>
<td>0</td>
<td>Endpoint of contour[i], ordinate value Protection zone for spindle chuck, contour element 7</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,8]$</td>
<td>0</td>
<td>Endpoint of contour[i], ordinate value Protection zone for spindle chuck, contour element 8</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,9]$</td>
<td>0</td>
<td>Endpoint of contour[i], ordinate value Protection zone for spindle chuck, contour element 9</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[0,0]$</td>
<td>0</td>
<td>Endpoint of contour[i], abscissa value Protection zone for spindle chuck, contour element 0</td>
</tr>
<tr>
<td>System variable</td>
<td>Value</td>
<td>Remark</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[0,1]$</td>
<td>110</td>
<td>; Endpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 1</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[0,2]$</td>
<td>110</td>
<td>; Endpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 2</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[0,3]$</td>
<td>0</td>
<td>; Endpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 3</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[0,4]$</td>
<td>0</td>
<td>; Endpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 4</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[0,5]$</td>
<td>0</td>
<td>; Endpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 5</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[0,6]$</td>
<td>0</td>
<td>; Endpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 6</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[0,7]$</td>
<td>0</td>
<td>; Endpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 7</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[0,8]$</td>
<td>0</td>
<td>; Endpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 8</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[0,9]$</td>
<td>0</td>
<td>; Endpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 9</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,0]$</td>
<td>0</td>
<td>; Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 0</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,1]$</td>
<td>0</td>
<td>; Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 1</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,2]$</td>
<td>0</td>
<td>; Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 2</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,3]$</td>
<td>0</td>
<td>; Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 3</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,4]$</td>
<td>0</td>
<td>; Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 4</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,5]$</td>
<td>0</td>
<td>; Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 5</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,6]$</td>
<td>0</td>
<td>; Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 6</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,7]$</td>
<td>0</td>
<td>; Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 7</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,8]$</td>
<td>0</td>
<td>; Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 8</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,9]$</td>
<td>0</td>
<td>; Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 9</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0,0]$</td>
<td>0</td>
<td>; Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 0</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0,1]$</td>
<td>0</td>
<td>; Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 1</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0,2]$</td>
<td>0</td>
<td>; Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 2</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0,3]$</td>
<td>0</td>
<td>; Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Protection zone for spindle chuck, contour element 3</td>
</tr>
</tbody>
</table>
### Table 3-3 Protection zone: Workpiece and toolholder

<table>
<thead>
<tr>
<th>System variable</th>
<th>Value</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SN_PA_ACTIV_IMMED[0]$</td>
<td>0</td>
<td>Protection zone for workpiece not immediately active</td>
</tr>
<tr>
<td>$SN_PA_ACTIV_IMMED[1]$</td>
<td>0</td>
<td>Protection zone for toolholder not immediately active</td>
</tr>
<tr>
<td>$SC_PA_TW[0]$</td>
<td>&quot; &quot;</td>
<td>Protection zone for workpiece, channel-specific</td>
</tr>
<tr>
<td>$SC_PA_TW[1]$</td>
<td>'H01'</td>
<td>Protection zone for toolholder, channel-specific</td>
</tr>
<tr>
<td>$SC_PA_ORI[0]$</td>
<td>1</td>
<td>Orientation of the protection zone: 1 = 3. and first geometry axis</td>
</tr>
<tr>
<td>$SC_PA_ORI[1]$</td>
<td>1</td>
<td>Orientation of the protection zone: 1 = 3. and first geometry axis</td>
</tr>
<tr>
<td>$SC_PA_LIM_3DIM[0]$</td>
<td>0</td>
<td>Type of limitation in the third dimension: 0 = no limitation</td>
</tr>
<tr>
<td>$SC_PA_LIM_3DIM[1]$</td>
<td>0</td>
<td>Type of limitation in the third dimension: 0 = no limitation</td>
</tr>
<tr>
<td>$SC_PA_PLUS_LIM[0]$</td>
<td>0</td>
<td>Value of limitation in positive direction in the third dimension</td>
</tr>
<tr>
<td>$SC_PA_PLUS_LIM[1]$</td>
<td>0</td>
<td>Value of limitation in positive direction in the third dimension</td>
</tr>
<tr>
<td>$SC_PA_MINUS_LIM[0]$</td>
<td>0</td>
<td>Value of limitation in negative direction in the third dimension</td>
</tr>
<tr>
<td>$SC_PA_MINUS_LIM[1]$</td>
<td>0</td>
<td>Value of limitation in negative direction in the third dimension</td>
</tr>
<tr>
<td>$SC_PA_CONT_NUM[0]$</td>
<td>4</td>
<td>Number of valid contour elements</td>
</tr>
<tr>
<td>$SC_PA_CONT_NUM[1]$</td>
<td>5</td>
<td>Number of valid contour elements</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,0]$</td>
<td>1</td>
<td>Contour type[i] : 1 = G1 for even</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,1]$</td>
<td>1</td>
<td>Contour type[i] : 1 = G1 for even</td>
</tr>
<tr>
<td>System variable</td>
<td>Value</td>
<td>Remark</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,2]</td>
<td>1</td>
<td>Contour type[i] : 1 = G1 for even</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 2</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,3]</td>
<td>1</td>
<td>Contour type[i] : 1 = G1 for even</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 3</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,4]</td>
<td>1</td>
<td>Contour type[i] : 1 = G1 for even</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 4</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,5]</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 5</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,6]</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 6</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,7]</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 7</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,8]</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 8</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[0,9]</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 9</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[1,0]</td>
<td>1</td>
<td>Contour type[i] : 1 = G1 for even</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 0</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[1,1]</td>
<td>3</td>
<td>Contour type[i] : 3 = G3 for circuit element, counter-clockwise</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 1</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[1,2]</td>
<td>1</td>
<td>Contour type[i] : 1 = G1 for even</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 2</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[1,3]</td>
<td>1</td>
<td>Contour type[i] : 1 = G1 for even</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 3</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[1,4]</td>
<td>1</td>
<td>Contour type[i] : 1 = G1 for even</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 4</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[1,5]</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 5</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[1,6]</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 6</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[1,7]</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 7</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[1,8]</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 8</td>
</tr>
<tr>
<td>$SN_PA_CONT_TYP[1,9]</td>
<td>0</td>
<td>Contour type[i] : 0 = not defined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 9</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,0]</td>
<td>-80</td>
<td>Endpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 0</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,1]</td>
<td>-80</td>
<td>Endpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 1</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,2]</td>
<td>80</td>
<td>Endpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 2</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,3]</td>
<td>80</td>
<td>Endpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 3</td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD[0,4]</td>
<td>0</td>
<td>Endpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 4</td>
</tr>
</tbody>
</table>
### System variable

<table>
<thead>
<tr>
<th>System variable</th>
<th>Value</th>
<th>Remark</th>
</tr>
</thead>
</table>
| $SN_PA_CONT_ORD[0,5]$ | 0 | Endpoint of contour[i], ordinate value  
Protection zone for workpiece, contour element 5 |
| $SN_PA_CONT_ORD[0,6]$ | 0 | Endpoint of contour[i], ordinate value  
Protection zone for workpiece, contour element 6 |
| $SN_PA_CONT_ORD[0,7]$ | 0 | Endpoint of contour[i], ordinate value  
Protection zone for workpiece, contour element 7 |
| $SN_PA_CONT_ORD[0,8]$ | 0 | Endpoint of contour[i], ordinate value  
Protection zone for workpiece, contour element 8 |
| $SN_PA_CONT_ORD[0,9]$ | 0 | Endpoint of contour[i], ordinate value  
Protection zone for workpiece, contour element 9 |
| $SN_PA_CONT_ORD[1,0]$ | -190 | Endpoint of contour[i], ordinate value  
Protection zone for toolholder, contour element 0 |
| $SN_PA_CONT_ORD[1,1]$ | -210 | Endpoint of contour[i], ordinate value  
Protection zone for toolholder, contour element 1 |
| $SN_PA_CONT_ORD[1,2]$ | -210 | Endpoint of contour[i], ordinate value  
Protection zone for toolholder, contour element 2 |
| $SN_PA_CONT_ORD[1,3]$ | 0 | Endpoint of contour[i], ordinate value  
Protection zone for toolholder, contour element 3 |
| $SN_PA_CONT_ORD[1,4]$ | 0 | Endpoint of contour[i], ordinate value  
Protection zone for toolholder, contour element 4 |
| $SN_PA_CONT_ORD[1,5]$ | 0 | Endpoint of contour[i], ordinate value  
Protection zone for toolholder, contour element 5 |
| $SN_PA_CONT_ORD[1,6]$ | 0 | Endpoint of contour[i], ordinate value  
Protection zone for toolholder, contour element 6 |
| $SN_PA_CONT_ORD[1,7]$ | 0 | Endpoint of contour[i], ordinate value  
Protection zone for toolholder, contour element 7 |
| $SN_PA_CONT_ORD[1,8]$ | 0 | Endpoint of contour[i], ordinate value  
Protection zone for toolholder, contour element 8 |
| $SN_PA_CONT_ORD[1,9]$ | 0 | Endpoint of contour[i], ordinate value  
Protection zone for toolholder, contour element 9 |
| $SN_PA_CONT_ABS[0,0]$ | 0 | Endpoint of contour[i], abscissa value  
Protection zone for workpiece, contour element 0 |
| $SN_PA_CONT_ABS[0,1]$ | 40 | Endpoint of contour[i], abscissa value  
Protection zone for workpiece, contour element 1 |
| $SN_PA_CONT_ABS[0,2]$ | 40 | Endpoint of contour[i], abscissa value  
Protection zone for workpiece, contour element 2 |
| $SN_PA_CONT_ABS[0,3]$ | 0 | Endpoint of contour[i], abscissa value  
Protection zone for workpiece, contour element 3 |
| $SN_PA_CONT_ABS[0,4]$ | -50 | Endpoint of contour[i], abscissa value  
Protection zone for workpiece, contour element 4 |
| $SN_PA_CONT_ABS[0,5]$ | 0 | Endpoint of contour[i], abscissa value  
Protection zone for workpiece, contour element 5 |
| $SN_PA_CONT_ABS[0,6]$ | 0 | Endpoint of contour[i], abscissa value  
Protection zone for workpiece, contour element 6 |
| $SN_PA_CONT_ABS[0,7]$ | 0 | Endpoint of contour[i], abscissa value  
Protection zone for workpiece, contour element 7 |
<table>
<thead>
<tr>
<th>System variable</th>
<th>Value</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SN_PA_CONT_ABS[0,8]$</td>
<td>0</td>
<td>Endpoint of contour[i], abscissa value; Protection zone for workpiece, contour element 8</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[0,9]$</td>
<td>0</td>
<td>Endpoint of contour[i], abscissa value; Protection zone for workpiece, contour element 9</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[1,0]$</td>
<td>-50</td>
<td>Endpoint of contour[i], abscissa value; Protection zone for toolholder, contour element 0</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[1,1]$</td>
<td>-30</td>
<td>Endpoint of contour[i], abscissa value; Protection zone for toolholder, contour element 1</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[1,2]$</td>
<td>20</td>
<td>Endpoint of contour[i], abscissa value; Protection zone for toolholder, contour element 2</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[1,3]$</td>
<td>50</td>
<td>Endpoint of contour[i], abscissa value; Protection zone for toolholder, contour element 3</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[1,4]$</td>
<td>-50</td>
<td>Endpoint of contour[i], abscissa value; Protection zone for toolholder, contour element 4</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[1,5]$</td>
<td>0</td>
<td>Endpoint of contour[i], abscissa value; Protection zone for toolholder, contour element 5</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[1,6]$</td>
<td>0</td>
<td>Endpoint of contour[i], abscissa value; Protection zone for toolholder, contour element 6</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[1,7]$</td>
<td>0</td>
<td>Endpoint of contour[i], abscissa value; Protection zone for toolholder, contour element 7</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[1,8]$</td>
<td>0</td>
<td>Endpoint of contour[i], abscissa value; Protection zone for toolholder, contour element 8</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS[1,9]$</td>
<td>0</td>
<td>Endpoint of contour[i], abscissa value; Protection zone for toolholder, contour element 9</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,0]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value; Protection zone for workpiece, contour element 0</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,1]$</td>
<td>-190</td>
<td>Midpoint of contour[i], ordinate value; Protection zone for workpiece, contour element 1</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,2]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value; Protection zone for workpiece, contour element 2</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,3]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value; Protection zone for workpiece, contour element 3</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,4]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value; Protection zone for workpiece, contour element 4</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,5]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value; Protection zone for workpiece, contour element 5</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,6]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value; Protection zone for workpiece, contour element 6</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,7]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value; Protection zone for workpiece, contour element 7</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,8]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value; Protection zone for workpiece, contour element 8</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[0,9]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value; Protection zone for toolholder, contour element 9</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[1,0]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value; Protection zone for toolholder, contour element 0</td>
</tr>
<tr>
<td>System variable</td>
<td>Value</td>
<td>Remark</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[1.1]$</td>
<td>-190</td>
<td>Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 1</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[1.2]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 2</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[1.3]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 3</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[1.4]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 4</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[1.5]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 5</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[1.6]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 6</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[1.7]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 7</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[1.8]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 8</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD[1.9]$</td>
<td>0</td>
<td>Midpoint of contour[i], ordinate value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 9</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0,0]$</td>
<td>0</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 0</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0.1]$</td>
<td>-30</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 1</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0.2]$</td>
<td>0</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 2</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0.3]$</td>
<td>0</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 3</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0.4]$</td>
<td>0</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 4</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0.5]$</td>
<td>0</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 5</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0.6]$</td>
<td>0</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 6</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0.7]$</td>
<td>0</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 7</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0.8]$</td>
<td>0</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 8</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[0.9]$</td>
<td>0</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for workpiece, contour element 9</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[1.0]$</td>
<td>0</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 0</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[1.1]$</td>
<td>-30</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 1</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[1.2]$</td>
<td>0</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 2</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS[1.3]$</td>
<td>0</td>
<td>Midpoint of contour[i], abscissa value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protection zone for toolholder, contour element 3</td>
</tr>
</tbody>
</table>
### Activation

Table 3-4 Part program excerpt for activating the three protection zones for spindle chuck, workpiece, and toolholder:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPROT(1, 2, 0, 0, 0)</td>
<td>Protection zone: Spindle chuck</td>
</tr>
<tr>
<td>CPROT(1, 2, 0, 0, 100)</td>
<td>Protection zone: Workpiece with 100 mm offset in the Z axis</td>
</tr>
<tr>
<td>CPROT(2, 2, 0, 0, 0)</td>
<td>Protection zone: Toolholder</td>
</tr>
</tbody>
</table>

### 3.6 Data lists

#### 3.6.1 Machine data

#### 3.6.1.1 NC-specific machine data

### Axis monitoring

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10604</td>
<td>WALIM_GEOAX_CHANGE_MODE</td>
<td>Working area limitation during switchover of geometry axes</td>
</tr>
<tr>
<td>10710</td>
<td>PROG_SD_RESET_SAVE_TAB</td>
<td>Setting data to be updated</td>
</tr>
</tbody>
</table>
### Protection zones

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10618</td>
<td>PROTAREA_GEOAX_CHANGE_MODE</td>
<td>Protection zone for switchover of geo axes</td>
</tr>
<tr>
<td>18190</td>
<td>MM_NUM_PROTECT_AREA_NCK</td>
<td>Number of files for machinerelated protection zones</td>
</tr>
</tbody>
</table>

### Axespecific machine data

#### Axis monitoring functions

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20150</td>
<td>GCODE_RESET_VALUES</td>
<td>Initial setting of the G groups</td>
</tr>
<tr>
<td>21020</td>
<td>WORKAREA_WITH_TOOL_RADIUS</td>
<td>Allowance for tool radius with working area limitation</td>
</tr>
<tr>
<td>24130</td>
<td>TRAFO_INCLUDES_TOOL_1</td>
<td>Tool handling with active transformation 1.</td>
</tr>
<tr>
<td>24230</td>
<td>TRAFO_INCLUDES_TOOL_2</td>
<td>Tool handling with active transformation 2.</td>
</tr>
<tr>
<td>24330</td>
<td>TRAFO_INCLUDES_TOOL_3</td>
<td>Tool handling with active transformation 3.</td>
</tr>
<tr>
<td>24426</td>
<td>TRAFO_INCLUDES_TOOL_4</td>
<td>Tool handling with active transformation 4.</td>
</tr>
<tr>
<td>24436</td>
<td>TRAFO_INCLUDES_TOOL_5</td>
<td>Tool handling with active transformation 5.</td>
</tr>
<tr>
<td>24446</td>
<td>TRAFO_INCLUDES_TOOL_6</td>
<td>Tool handling with active transformation 6.</td>
</tr>
<tr>
<td>24456</td>
<td>TRAFO_INCLUDES_TOOL_7</td>
<td>Tool handling with active transformation 7.</td>
</tr>
<tr>
<td>24466</td>
<td>TRAFO_INCLUDES_TOOL_8</td>
<td>Tool handling with active transformation 8.</td>
</tr>
<tr>
<td>24476</td>
<td>TRAFO_INCLUDES_TOOL_9</td>
<td>Tool handling with active transformation 9.</td>
</tr>
<tr>
<td>24486</td>
<td>TRAFO_INCLUDES_TOOL_10</td>
<td>Tool handling with active transformation 10.</td>
</tr>
<tr>
<td>25106</td>
<td>TRAFO_INCLUDES_TOOL_11</td>
<td>Tool handling with active transformation 11.</td>
</tr>
<tr>
<td>25116</td>
<td>TRAFO_INCLUDES_TOOL_12</td>
<td>Tool handling with active transformation 12.</td>
</tr>
<tr>
<td>25126</td>
<td>TRAFO_INCLUDES_TOOL_13</td>
<td>Tool handling with active transformation 13.</td>
</tr>
<tr>
<td>25136</td>
<td>TRAFO_INCLUDES_TOOL_14</td>
<td>Tool handling with active transformation 14.</td>
</tr>
<tr>
<td>25146</td>
<td>TRAFO_INCLUDES_TOOL_15</td>
<td>Tool handling with active transformation 15.</td>
</tr>
<tr>
<td>25156</td>
<td>TRAFO_INCLUDES_TOOL_16</td>
<td>Tool handling with active transformation 16.</td>
</tr>
<tr>
<td>25166</td>
<td>TRAFO_INCLUDES_TOOL_17</td>
<td>Tool handling with active transformation 17.</td>
</tr>
<tr>
<td>25176</td>
<td>TRAFO_INCLUDES_TOOL_18</td>
<td>Tool handling with active transformation 18.</td>
</tr>
<tr>
<td>25186</td>
<td>TRAFO_INCLUDES_TOOL_19</td>
<td>Tool handling with active transformation 19.</td>
</tr>
<tr>
<td>25196</td>
<td>TRAFO_INCLUDES_TOOL_20</td>
<td>Tool handling with active transformation 20.</td>
</tr>
<tr>
<td>28600</td>
<td>MM_NUM_WORKAREA_CS_GROUPS</td>
<td>Number of coordinate system-specific working area limitations</td>
</tr>
</tbody>
</table>
### Protection zones

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC___</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>28200</td>
<td>MM_NUM_PROTECT_AREA_CHAN (SRAM)</td>
<td>Number of files for channel-specific protection zones</td>
</tr>
<tr>
<td>28210</td>
<td>MM_NUM_PROTECT_AREA_ACTIVE</td>
<td>Number of simultaneously active protection zones in one channel</td>
</tr>
<tr>
<td>28212</td>
<td>MM_NUM_PROTECT_AREA_CONTUR</td>
<td>Elements for active protection zones (DRAM)</td>
</tr>
</tbody>
</table>

### 3.6.1.3 Axis/spindlespecific machine data

### Axis monitoring functions

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA___</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>30200</td>
<td>NUM_ENCS</td>
<td>Number of encoders</td>
</tr>
<tr>
<td>30240</td>
<td>ENC_TYPE</td>
<td>Encoder type of the actual value sensing (position actual value)</td>
</tr>
<tr>
<td>30310</td>
<td>ROT_IS_MODULO</td>
<td>Modulo conversion for rotary axis/spindle</td>
</tr>
<tr>
<td>30800</td>
<td>WORK_AREA_CHECK_TYPE</td>
<td>Type of checking of working area limits</td>
</tr>
<tr>
<td>31046</td>
<td>ENC_PASSIVE_PARKING</td>
<td>Parking the passive position measuring system</td>
</tr>
<tr>
<td>32200</td>
<td>POSCTRL_GAIN [n]</td>
<td>Servo gain factor (K_v)</td>
</tr>
<tr>
<td>32250</td>
<td>RATED_OUTVAL</td>
<td>Rated output voltage</td>
</tr>
<tr>
<td>32260</td>
<td>RATED_VELO</td>
<td>Rated motor speed</td>
</tr>
<tr>
<td>32300</td>
<td>MAX_AX_ACCEL</td>
<td>Maximum axis acceleration</td>
</tr>
<tr>
<td>32800</td>
<td>EQUIV_CURRCTRL_TIME</td>
<td>Equivalent time constant current control loop for feedforward control</td>
</tr>
<tr>
<td>32810</td>
<td>EQUIV_SPEEDCTRL_TIME</td>
<td>Equivalent time constant speed control loop for feedforward control</td>
</tr>
<tr>
<td>32910</td>
<td>DYN_MATCH_TIME [n]</td>
<td>Time constant for dynamic response adaptation</td>
</tr>
<tr>
<td>34210</td>
<td>ENC_REFP_STATE</td>
<td>Encoder status</td>
</tr>
<tr>
<td>35160</td>
<td>SPIND_EXTERN_VELO_LIMIT</td>
<td>Spindle speed limitation via PLCC</td>
</tr>
<tr>
<td>36000</td>
<td>STOP_LIMIT_COARSE</td>
<td>Exact stop coarse</td>
</tr>
<tr>
<td>36010</td>
<td>STOP_LIMIT_FINE</td>
<td>Exact stop fine</td>
</tr>
<tr>
<td>36020</td>
<td>POSITIONING_TIME</td>
<td>Delay time exact stop fine</td>
</tr>
<tr>
<td>36030</td>
<td>STANDSTILL_POS_TOL</td>
<td>Zero speed tolerance</td>
</tr>
<tr>
<td>36040</td>
<td>STANDSTILL_DELAY_TIME</td>
<td>Delay time zero-speed monitoring</td>
</tr>
<tr>
<td>36050</td>
<td>CLAMP_POS_TOL</td>
<td>Clamping tolerance with IS &quot;Clamping active&quot;</td>
</tr>
<tr>
<td>36052</td>
<td>STOP_ON_CLAMPING</td>
<td>Special functions for clamped axis</td>
</tr>
<tr>
<td>36060</td>
<td>STANDSTILL_VELO_TOL</td>
<td>Maximum velocity/speed &quot;Axis/spindle stationary&quot;</td>
</tr>
<tr>
<td>36100</td>
<td>POS_LIMIT_MINUS</td>
<td>1. software limit switch minus</td>
</tr>
<tr>
<td>36110</td>
<td>POS_LIMIT_PLUS</td>
<td>1. software limit switch plus</td>
</tr>
<tr>
<td>36120</td>
<td>POS_LIMIT_MINUS2</td>
<td>2. software limit switch minus</td>
</tr>
<tr>
<td>36130</td>
<td>POS_LIMIT_PLUS2</td>
<td>2. software limit switch plus</td>
</tr>
<tr>
<td>36200</td>
<td>AX_VELO_LIMIT</td>
<td>Threshold value for velocity monitoring</td>
</tr>
<tr>
<td>36210</td>
<td>CTRLOUT_LIMIT</td>
<td>Maximum speed setpoint</td>
</tr>
<tr>
<td>36220</td>
<td>CTRLOUT_LIMIT_TIME</td>
<td>Delay time for speed-setpoint monitoring</td>
</tr>
</tbody>
</table>
### 3.6.2 Setting data

#### 3.6.2.1 Axis/spindlespecific setting data

**Axis monitoring functions**

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $SA__</th>
</tr>
</thead>
<tbody>
<tr>
<td>43400</td>
<td>WORKAREA_PlUS_ENABLE</td>
</tr>
<tr>
<td>43410</td>
<td>WORKAREA_MINUS_ENABLE</td>
</tr>
<tr>
<td>43420</td>
<td>WORKAREA_LIMIT_PLUS</td>
</tr>
<tr>
<td>43430</td>
<td>WORKAREA_LIMIT_MINUS</td>
</tr>
</tbody>
</table>

**Protection zones**

None

---

### 3.6.3 Signals

#### 3.6.3.1 Signals to channel

**Axis monitoring functions**

None
Protection zones

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable protection zones</td>
<td>DB21, ..., DBX1.1</td>
<td>DB320x.DBX1.1</td>
</tr>
<tr>
<td>Feed disable</td>
<td>DB21, ..., DBX6.0</td>
<td>DB320x.DBX6.0</td>
</tr>
<tr>
<td>Activate machinerelated protection zones 1-8</td>
<td>DB21, ..., DBX8.0-7</td>
<td>DB320x.DBX8.0-7</td>
</tr>
<tr>
<td>Activate machinerelated protection zone 9</td>
<td>DB21, ..., DBX9.0</td>
<td>DB320x.DBX9.0</td>
</tr>
<tr>
<td>Activate machinerelated protection zone 10</td>
<td>DB21, ..., DBX9.1</td>
<td>DB320x.DBX9.1</td>
</tr>
<tr>
<td>Activate channelspecific protection zone 1-8</td>
<td>DB21, ..., DBX10.0-7</td>
<td>DB320x.DBX10.0-7</td>
</tr>
<tr>
<td>Activate channelspecific protection zone 9</td>
<td>DB21, ..., DBX11.0</td>
<td>DB320x.DBX11.0</td>
</tr>
<tr>
<td>Activate channelspecific protection zone 10</td>
<td>DB21, ..., DBX11.1</td>
<td>DB320x.DBX11.1</td>
</tr>
</tbody>
</table>

3.6.3.2 Signals from channel

Axis monitoring functions
None

Protection zones

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machinerelated protection zones 1-8 preactivated</td>
<td>DB21, ..., DBX272.0-7</td>
<td>DB330x.DBX8.0-7</td>
</tr>
<tr>
<td>Machinerelated protection zone 9 preactivated</td>
<td>DB21, ..., DBX273.0</td>
<td>DB330x.DBX9.0</td>
</tr>
<tr>
<td>Machinerelated protection zone 10 preactivated</td>
<td>DB21, ..., DBX273.1</td>
<td>DB330x.DBX9.1</td>
</tr>
<tr>
<td>Channelspecific protection zones 1-8 preactivated</td>
<td>DB21, ..., DBX274.0-7</td>
<td>DB330x.DBX10.0-7</td>
</tr>
<tr>
<td>Channelspecific protection zone 9 preactivated</td>
<td>DB21, ..., DBX275.0</td>
<td>DB330x.DBX11.0</td>
</tr>
<tr>
<td>Channelspecific protection zone 10 preactivated</td>
<td>DB21, ..., DBX275.1</td>
<td>DB330x.DBX11.1</td>
</tr>
<tr>
<td>Machinerelated protection zones 1-8 violated</td>
<td>DB21, ..., DBX276.0-7</td>
<td>DB330x.DBX12.0-7</td>
</tr>
<tr>
<td>Machinerelated protection zone 9 violated</td>
<td>DB21, ..., DBX277.0</td>
<td>DB330x.DBX13.0</td>
</tr>
<tr>
<td>Machinerelated protection zone 10 violated</td>
<td>DB21, ..., DBX277.1</td>
<td>DB330x.DBX13.1</td>
</tr>
<tr>
<td>Channelspecific protection zones 1-8 violated</td>
<td>DB21, ..., DBX278.0-7</td>
<td>DB330x.DBX14.0-7</td>
</tr>
<tr>
<td>Channelspecific protection zone 9 violated</td>
<td>DB21, ..., DBX279.0</td>
<td>DB330x.DBX15.0</td>
</tr>
<tr>
<td>Channelspecific protection zone 10 violated</td>
<td>DB21, ..., DBX279.1</td>
<td>DB330x.DBX15.1</td>
</tr>
</tbody>
</table>

3.6.3.3 Signals to axis/spindle

Axis monitoring functions

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow-up mode</td>
<td>DB31, ..., DBX1.4</td>
<td>DB380x.DBX1.4</td>
</tr>
<tr>
<td>Position measuring system 1 / 2</td>
<td>DB31, ..., DBX1.5/6</td>
<td>DB380x.DBX1.5/6</td>
</tr>
</tbody>
</table>
A3: Axis Monitoring, Protection Zones

3.6 Data lists

### Protection zones

None

#### 3.6.3.4 Signals from axis/spindle

**Axis monitoring functions**

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder limit frequency exceeded 1 / 2</td>
<td>DB31, ... DBX60.2/3</td>
<td>DB390x.DBX0.2</td>
</tr>
<tr>
<td>Referenced/synchronized 1/2</td>
<td>DB31, ... DBX60.4/5</td>
<td>DB390x.DBX0.4/5</td>
</tr>
<tr>
<td>Travel command minus/plus</td>
<td>DB31, ... DBX64.6/7</td>
<td>DB390x.DBX4.6/7</td>
</tr>
<tr>
<td>Position measuring system 1/2 activated</td>
<td>DB31, ... DBX102.5/6</td>
<td>DB390x.DBX5006.5/6</td>
</tr>
</tbody>
</table>

**Protection zones**

None
4.1 Brief Description

Exact stop or exact stop mode

In exact stop traversing mode, all axes involved in the traversing motion (except axes of modal traversing modes) are decelerated at the end of each block until they come to a standstill. The transition to the next block occurs only when all axes involved in the traversing motion have reached their programmed target position with subject to the selected exact stop criterion.

Continuous-path mode

In continuous-path mode, the NC attempts to keep the programmed path velocity as constant as possible. In particular, deceleration of the path axes at the block limits of the part program is to be avoided.

LookAhead

LookAhead is a function for optimizing the continuous path mode.

Smooth machining of workpieces is necessary to ensure a high-quality surface finish. For this reason, path velocity variations should be avoided during machining whenever possible. Without LookAhead, the NC only takes the traversing block immediately following the current traversing block into consideration when determining the possible path velocity. If the following block contains only a short path, the NC must reduce the path velocity (decelerate in the current block) to be able to stop in time at the end of the next block, if necessary.

When the NC "looks ahead" over a configurable number of traversing blocks following the current traversing block, a much higher path velocity can be attained under certain circumstances because the NC now has considerably more traversing blocks and more path available for calculation.

This results in the following advantages:

- Machining with higher path velocities on average
- Improved surface quality by avoiding deceleration and acceleration processes

Smoothing the path velocity

"Smoothing the path velocity" is a function especially for applications (such as high speed milling in mold and die production) that require an extremely steady path velocity. Deceleration and acceleration processes that would cause high-frequency excitations of machine resonances are avoided with the "Smoothing the path velocity" function.
This results in the following advantages:

- Improved surface quality and machining time by avoiding excitation of machine resonances.
- Constant profile of path velocity and cutting rates by avoiding "unnecessary" acceleration processes, i.e. acceleration processes that do not greatly improve the program run time.

**Adaptation of the dynamic path response**

In addition to "smoothing the path velocity", "dynamic path response adaptation" is another function for avoiding high-frequency excitations of machine resonances while optimizing the dynamic path response at the same time. To this end, highly frequent changes in path velocity are automatically executed with lower jerk or acceleration values than the dynamic response limit value parameters assigned in the machine data.

Thus, with low-frequency changes in path velocity, the full dynamic response limit values apply, whereas with high-frequency changes, only the reduced dynamic response limit values act due to the automatic dynamic response adaptation.

**Dynamic response mode for path interpolation**

Optimizing the path dynamic response also includes the technology-specific dynamic response settings which are preset for different processing technologies (including tapping, roughing, smoothing) and can be activated in the part program by calling the respective dynamic response mode.

**Free-form surface mode**

Any fluctuation in curvature or torsion leads to a change in path velocity. This generally results in unnecessary decelerating and accelerating during the processing of free-form surface workpieces, which may adversely affect the quality of the surfaces of the workpieces.

The following functions are available for processing free-form surfaces.

- "Free-form surface mode: Basic functions"
  This makes the definition of the path velocity profile "less sensitive" to fluctuations in curvature and torsion.

- "Free-form surface mode: Extension function"
  This extension in standard LookAhead functionality is used to calculate the path velocity profile during the processing of free-form surfaces.

The advantages of free-form surface mode lie in a more homogeneous workpiece surface and lower machine load.

**NC block compression**

When a workpiece design is completed with a CAD/CAM system, the CAD/CAM system generally also compiles the corresponding part program to create the workpiece surface. To do so, most CAD/CAM systems use linear blocks to describe even curved sections of the workpiece surface. Many interpolation points are generally necessary to maintain the required contour accuracy. This results in many linear blocks, typically with very short paths.
The "NC block compressor" function uses polynomial blocks to perform a subsequent approximation of the contour specified by the linear blocks. During this process, an assignable number of linear blocks is replaced by a polynomial block. Furthermore, the number of linear blocks that can be replaced by a polynomial block also depends on the specified maximum permissible contour deviation and the contour profile.

Use of polynomial blocks provides the following advantages:

- Reduction of the number of required part program blocks for describing the workpiece contour
- Higher maximum path velocities

**Combine short spline blocks**

A spline defines a curve which is formed from 2nd or 3rd degree polynomials. With spline interpolation, the controller can generate a smooth curve characteristic from only a few specified interpolation points of a set contour.

**References:**
Programming Manual, Advanced, Special motion commands, chapter: Spline interpolation

The advantages of the spline interpolation as compared to the linear interpolation are:

- Reduction of the number of required part program blocks for describing a curved contour.
- Soft, mechanical system-limiting curve characteristic also during transition between the part program blocks.

The disadvantages of the spline interpolation as compared to the linear interpolation are:

- For a spline curve no exact curve characteristic, but only a tolerance band can be specified, within which the spline curve should lie.

As with linear interpolation, the processing of splines can produce such short blocks that the path velocity must be reduced to enable interpolation of the spline blocks. This is also the case, when the spline actually has a long, smooth curve. The "Combine short spline blocks" function allows you to combine these spline blocks such that the resulting block length is sufficient and does not reduce the path velocity.

**Note**

**NC block compressor**

The NC block compressor (COMPON, COMPCURV or COMPCAD) cannot be employed while compressing spline blocks since it can only be used to compress linear blocks.
4.2 Exact stop mode

Exact stop or exact stop mode

In exact stop traversing mode, all path axes and special axes involved in the traversing motion that are not traversed modally, are decelerated at the end of each block until they come to a standstill. The transition to the next block occurs only when all axes involved in the traversing motion have reached their programmed target position with subject to the selected exact stop criterion.

This results in the following response:

- The program run time is considerably longer compared to continuous path mode due to the deceleration of the axes and the wait time until "Exact stop" status is reached for all machine axes involved.
- In exact stop mode, undercuts can occur on the workpiece surface during machining.

Status "Exact stop"

The state of the machine axis based on the position difference relative to its position setpoint at the end of a traversing motion is also called an exact stop. The machine axis reaches the "exact stop" state, as soon as its following error is less than the specified position difference (exact stop criterion).

Application

Exact stop mode should always be used when the programmed contour must be executed exactly.

Activation

Exact stop mode can be activated on a modal basis or in blocks in the part program:

<table>
<thead>
<tr>
<th>G command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>G60</td>
<td>Exact stop with modal effect</td>
</tr>
<tr>
<td>G9</td>
<td>Exact stop with block-by-block effect</td>
</tr>
</tbody>
</table>
Exact stop criteria "Exact stop coarse" and "Exact stop fine".

The exact stop criteria "Exact stop coarse" and "Exact stop fine" are used to specify tolerance windows for a machine axis reaching the "exact stop" state.

![Diagram showing tolerance windows of exact stop criteria](image)

Figure 4-1  Tolerance windows of exact stop criteria

Parameters are assigned to the two exact stop criteria via the machine data:
- MD36010 $MA_STOP_LIMIT_FINE (exact stop fine)
- MD36000 $MA_STOP_LIMIT_COARSE (exact stop coarse)

**Note**
The tolerance windows of the exact stop criteria "Exact stop coarse" and "Exact stop fine" should be assigned in such a way that the following requirement is fulfilled:

"Exact stop coarse" > "Exact stop fine"

**Exact stop criterion "Interpolator End"**

In the case of the exact stop criterion "Interpolator end" the block change to the next traversing block takes place, as soon as all path axes and special axes involved in the traversing motion, which do not traverse extending up to block, have reached their position according to set point programmed in the block. That is, the interpolator has executed the block.

The actual position and the following error of the relevant machine axes are not taken into consideration for exact stop criterion "Interpolator end". Thus, depending on the dynamic response of the machine axes, this can result in a relatively large smoothing of the contour at the block changes in comparison to the exact stop criteria "Exact stop coarse" and "Exact stop fine".
Activation of an exact stop criterion

An exact stop criterion is activated in the part program by programming the appropriate G command:

<table>
<thead>
<tr>
<th>G command</th>
<th>Exact-stop criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>G601</td>
<td>Exact stop fine</td>
</tr>
<tr>
<td>G602</td>
<td>Exact stop coarse</td>
</tr>
<tr>
<td>G603</td>
<td>Interpolator end</td>
</tr>
</tbody>
</table>

Block change depending on exact-stop criteria

The figure below illustrates the block change timing in terms of the selected exact stop criterion.

![Figure 4-2 Block change accordance to selected exact stop criterion](image)

Evaluation factor for exact stop criteria

A parameter set-dependent evaluation of the exact stop criteria can be specified via the following axis-specific machine data:

MD36012 $MA_STOP_LIMIT_FACTOR (exact stop coarse/fine and standstill factor)

Applications:
- Adaptation of the positioning response to different mass ratios, such as after a gearshift
- Reduction in positioning time, depending on various machining states, such as roughing and finishing
Assignable specification of the active exact stop criterion

The active exact stop criterion can be permanently specified for the part program commands of the first G function group irrespective of the exact stop criterion programmed in the part program. This specification can be made independently for each of the following part program commands:

- Rapid traverse: G0
- Machining commands: G1, G2, G3, CIP, ASPLINE, BSPLINE, CSPLINE, POLY, G33, G34, G35, G331, G332, OEMIPO01, OEMIPO02, CT

The setting is done in a channel-specific manner via the following machine data:
MD20550 $MC_EXACT_POS_MODE (exact stop conditions for G0 and G1)

Coding

Each exact stop criterion is location-coded:

MD20550 $MC_EXACT_POS_MODE = <ZE>

- Ones position E: Rapid traverse
- Tens position Z: all other part program commands in the first G function group

<table>
<thead>
<tr>
<th>Z or E</th>
<th>Active exact stop criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Programmed exact stop criterion</td>
</tr>
<tr>
<td>1</td>
<td>G601 (Exact stop window fine)</td>
</tr>
<tr>
<td>2</td>
<td>G602 (Exact stop window coarse)</td>
</tr>
<tr>
<td>3</td>
<td>G603 (Interpolator end)</td>
</tr>
</tbody>
</table>

Example

MD20550 $MC_EXACT_POS_MODE = 02

Ones position = 2:

With rapid traverse, exact stop criterion G602 (Exact stop window coarse) is always active, irrespective of any programming in the parts program.

Tens digit = 0:

For traversing with all other parts program commands of the first G function group, the exact stop criterion programmed in the parts program is active.

Assignable exact stop criterion for rapid traverse transitions in continuous path mode

The behavior at the block transition of part program blocks before and after rapid traverse blocks can be parameterized as follows:

MD20552 $MC_EXACT_POS_MODE_G0_TO_G1 = <value>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No additional stop at the block transition.</td>
</tr>
<tr>
<td>1</td>
<td>Stop at block transition: Same behavior as in the case of G601 (Exact stop window fine)</td>
</tr>
<tr>
<td>2</td>
<td>Stop at block transition: Same behavior as G602 (Exact stop window coarse).</td>
</tr>
<tr>
<td>3</td>
<td>Stop at block transition: Same behavior as G603 (Interpolator end).</td>
</tr>
</tbody>
</table>
4.3 Continuous-path mode

4.3.1 General functionality

Continuous-path mode

In the continuous-path mode, the path velocity is not decelerated for the block change in order to permit the fulfillment of an exact stop criterion. The objective of this mode is to avoid rapid deceleration of the path axes at the block-change point so that the axis velocity remains as constant as possible when the program moves to the next block. To achieve this objective, the "LookAhead" function is also activated when the continuous-path mode is selected.

Continuous-path mode causes the smoothing and tangential shaping of angular block transitions by local changes in the programmed contour. The extent of the change relative to the programmed contour can be limited by specifying the overload factor or rounding criteria.

Continuous-path mode:
- Contour rounding.
- Reduces machining times by eliminating braking and acceleration processes that are required to fulfill the exact-stop criterion.
- Improves cutting conditions because of the more constant velocity.

Continuous-path mode is suitable if:
- A contour must be traversed as quickly as possible (e.g. with rapid traverse).
- The exact contour may deviate from the programmed contour within a specific tolerance for the purpose of obtaining a continuous contour.

Continuous-path mode is not suitable if:
- A contour is to be traversed precisely.
- An absolutely constant velocity is required.
Implicit exact stop

In some cases, an exact stop needs to be generated in continuous-path mode to allow the execution of subsequent actions. In such situations, the path velocity is reduced to zero.

- If auxiliary functions are output before the traverse motion, the previous block is only terminated when the selected exact-stop criterion is fulfilled.
- If auxiliary functions are to be output after the traverse motion, they are output after the interpolator end of the block.
- If an executable block (e.g. starting of a positioning axis) contains no travel information for the path axes, the previous block is terminated on reaching the selected exact-stop criterion.
- If a positioning axis is declared to be the geometry axis, the previous block is terminated at the interpolator end when the geometry axis is programmed.
- If a synchronized axis is programmed that was last programmed as a positioning axis or spindle (initial setting of the special axis is positioning axis), the previous block is ended at the interpolator end.
- If the transformation is changed, the block previously processed is terminated with the active exact-stop criterion.
- A block is terminated on interpolator end if the following block contains the changeover of the acceleration profile BRISK/SOFT (see Section "B2: Acceleration (Page 243)").
- If the "empty buffer" function is programmed, the previous block is terminated when the selected exact-stop criterion is reached.

Velocity = 0 in continuous-path mode

Regardless of the implicit exact stop response, the path motion is braked down to zero velocity at the end of the block in cases where:

- Positioning axes are programmed with the instruction POS, and their traversing time exceeds that of the path axes. The block change occurs when the "exact stop fine" of the positioning axes is reached.
- The time taken to position a spindle programmed with the instruction SPOS is longer than the traversing time of the path axes. The block change is carried out when the "exact stop fine" of the positioning spindle is reached.
- The current block contains traversing commands for geometry axes and the following block traversing commands for synchronized axes or, alternatively, the current block contains traversing commands for synchronized axes and the subsequent block traversing commands for geometry axes.
- Synchronization is required.
4.3.2 Velocity reduction according to overload factor

Function

The function lowers the path velocity in continuous-path mode until the non-tangential block transition can be traversed in one interpolation cycle while respecting the deceleration limit and taking an overload factor into account.

With the reduced velocity, axial jumps in velocity are produced with a non-tangential contour at the block transition. These jumps in velocity are also performed by the coupled motion synchronized axes. The jump in velocity prevents the path velocity dropping to zero. This jump is performed if the axial velocity was reduced with the axial acceleration to a velocity from which the new setpoint can be reached with the jump. The magnitude of the setpoint jump can be limited using an overload factor. Because the magnitude of the jump is axial, the minimum jump of the path axes which are active during the block change is considered during block transition.

Figure 4-3 Axial velocity change on block transition

With a practically tangential block transition, the path velocity is not reduced if the permissible axial accelerations are not exceeded. This means that very small bends in the contour (e.g. 0.5°) are overtraveled directly.
Overload factor

The overload factor restricts step changes in the machine axis velocity at block ends. To ensure that the velocity jump does not exceed the maximum load on the axis, the jump is derived from the acceleration of the axis.

The overload factor indicates the extent by which the acceleration of the machine axis (MD32300 $MA_MAX_AX ACCEL) may be exceeded for an IPO-cycle.

The velocity jump results as follows:
Velocity jump = axis acceleration * (overload factor-1) * interpolator cycle.

The overload factor is saved in the machine data:
MD32310 $MA_MAX ACCEL OVL_FACTOR (overload factor for axial velocity jumps)

Factor 1.0 means that only tangential transitions with finite velocity can be traversed. For all other transitions, the velocity is reduced to zero by changing the setpoint. This behavior is equivalent to the function "Exact stop with interpolator end". This is undesirable for continuous-path mode, so the factor must be set to greater than 1.0.

Note

For startup and installation, please note that the factor must be reduced if the machine is likely to be subject to vibrations during angular block transitions and rounding is not to be used.

By setting the following machine data, the block transitions are rounded independent of the set overload factor with G641/G642:
MD20490 $MC _IGNORE OVL_FACTOR FOR ADIS

Activation/deactivation

Continuous-path mode with a reduction in speed according to the overload factor can be activated in any NC part program block by the modal command G64.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode G64 can be deactivated by selecting:
- Modal exact stop G60
- Rounding G641, G642, G643, G644 or G645

Implicit continuous-path mode

If it is not possible to insert rounding blocks in continuous-path mode with rounding G641 due to the very short block path lengths (e.g. zero-clocked blocks), the mode is switched over to continuous-path mode G64.
4.3.3 Rounding

Function

The "Rounding" function adds intermediate blocks (positioning blocks) along a programmed contour (path axes) at non-continuous (angular) block transitions so that the resulting new block transition is continuous (tangential).

Synchronized axes

The rounding considers not only the geometry axes but also all synchronous axes. Although a continuous block transition cannot be created for both axis types concurrently for the parallel travel of path and synchronous axes. In this case, to favor path axes that always travel exactly, only an approximately continuous block transition is created for synchronous axes.

Rounding for G64

Rounding is performed even when to observe the dynamic-response limits at the block transition, a speed is required that for G64 exceeds the permitted speed at the block transition (see Section "Velocity reduction according to overload factor" (Page 186) "Overload factor").

Impact on synchronization conditions

The use of rounding shortens the programmed blocks between which the rounding block(s) are added. The programmed block boundary disappears and are then no longer available as criterion for any synchronization conditions (e.g. auxiliary function output parallel to motion, stop at block end).

Note

We recommend that when the "Rounding" function is used, synchronization conditions apply to the end of the block before the rounding location rather than the end of the inserted rounding block. The following block would then not be started and with a stop at the block end, the contour of the following block can still be changed manually.

Exceptions

In the following cases, no rounding occurs at the block transition, for example, between the N10 blocks after N20, i.e. no rounding block is added:

Implicit stopping of the traversing motion

Possible causes:

- Auxiliary function output active before the traversing motion for N20
- N20 does not contain any traversing motion for path axes
- In N20, an axis that was previously a positioning axis traverses as a path axis for the first time
- In N20, an axis that was previously a path axis traverses as a positioning axis for the first time
● Geometry axes traverse in N10 but not in N20
● Geometry axes traverse in N20 but not in N10
● Activation of thread-cutting G33 in N20
● Change from BRISK and SOFT
● Transformation-relevant axes are not completely assigned to the path motion (e.g. for oscillation axes, positioning axes).

The insertion of the rounding block would slow part program machining overproportionally

Possible causes:

● A program or program section consists of a number of very short traversal blocks (≈ 1 interpolator cycle / traversal block; because each traversal block requires at least one interpolator cycle, the inserted intermediate block would almost double the machining time)
● G64 (path control operation without rounding) without speed reduction active for block change
● The parameterized overload factor (MD32310 $MA_MAX_ACCEL_OVL_FACTOR) permits the traversal of the programmed contour without the path speed needing to be reduced.
See also: MD20490 $MC_IGNORE_OVL_FACTOR_FOR_ADIS

Path parameters inhibit the rounding

Possible causes:

● G641 (path control operation with rounding in accordance with the path criterion) is active but rapid traverse is active (G0) AND ADISPOS == 0 (rounding separation for G0)
● G641 (path control operation with rounding in accordance with the path criterion) is active but rapid traverse is not active AND ADIS == 0 (rounding separation for path functions G1, G2, G3, ...)
● G642 or G643 (path control operation with rounding while observing defined tolerances) is active but all tolerances == zero

N10 or N20 does not contain traversing motion (zero block).

Normally no zero blocks are created. Exceptions:

● Active synchronous action
● Program jumps
Impact on synchronization conditions

The programmed blocks between which the rounding contour is added are shortened during rounding. The original programmed block boundary disappears and is then no longer available for any synchronization conditions (e.g. auxiliary function output parallel to motion, stop at block end).

Note
We recommend that when the "Rounding" function is used, synchronization conditions apply to the end of the block before the rounding location rather than the end of the inserted rounding block. The following block would then not be started and with a stop at the block end, the contour of the following block can still be changed.

4.3.3.1 Rounding according to a path criterion (G641)

Function
In continuous-path mode with rounding according to a path criterion, the size of the rounding area is influenced by the path criteria ADIS and ADISPOS.

The path criteria ADIS and ADISPOS describe the maximum distances which a rounding block can occupy before and after a block.

Note
Acute angles produce rounding curves with a large degree of curvature and therefore cause a corresponding reduction in velocity.

Note
ADISPOS is programmed in the same way as ADIS, but must be used specifically for movements in rapid traverse mode G0.
Scope of the path criterion

- ADIS or ADISPOS must be programmed. If the default is "zero", G641 behaves like G64.
- If only one of the blocks involved is rapid traverse G0, the smaller rounding distance applies.
- If a very small value is used for ADIS, the controller must make sure that every interpolated block, even an intermediate rounding block, contains at least one interpolation point. The maximum path velocity is thereby limited to ADIS / interpolator cycle.
- Irrespective of ADIS and ADISPOS, the rounding area is limited by the block length. In blocks with short distances (distance < 4 * ADIS and < 4 * ADISPOS respectively), the rounding distance is reduced so that a traversable part of the original block is retained. The remaining length depends on the axis path and is approximately 60% of the distance still to be traversed in the block. ADIS or ADISPOS is therefore reduced to the remaining 40% of the distance to be traversed. This algorithm prevents a rounding block being inserted for a very small change in contour. In this case, switchover to continuous-path mode G64 is automatic until rounding blocks can be inserted again.

![Diagram](image)

Figure 4-4 Path with limitation of ADIS

Activation/deactivation

Continuous-path mode with rounding based on a path criterion can be activated in any NC part program block by the modal command G641. Before or on selection, the path criteria ADIS/ADISPOS must be specified.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode with rounding based on a path criterion (G641) can be deactivated by selecting:

- Modal exact stop (G60)
- Continuous-path mode G64, G642, G643, G644 or G645
4.3.3.2 Rounding in compliance with defined tolerances (G642/G643)

Function

In continuous-path mode involving rounding in compliance with defined tolerances, the rounding normally takes place while adhering to the maximum permissible path deviation. Instead of these axis-specific tolerances, the maintenance of the maximum contour deviation (contour tolerance) or the maximum angular deviation of the tool orientation (orientation tolerance) can be configured.

Activation

Continuous-path mode with rounding in compliance with defined tolerances can be activated in any NC part program block by the modal command G642 or G643.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode with rounding in compliance with defined tolerances (G642/G643) can be deactivated by selecting:

- Modal exact stop (G60)
- Continuous-path mode G64, G641, G644 or G645

Differences between G642 - G643

With regard to their rounding behavior, commands G642 and G643 differ as follows:

<table>
<thead>
<tr>
<th>G642</th>
<th>G643</th>
</tr>
</thead>
<tbody>
<tr>
<td>With G642, the rounding path is determined on the basis of the shortest distance for rounding all axes. This value is taken into account when generating a rounding block.</td>
<td>In the case of G643, each axis may have a different rounding path. The rounding travels are taken into account axis-specifically and block-internally (no separate rounding block).</td>
</tr>
<tr>
<td>With G642, the rounding area results from the smallest tolerance setting.</td>
<td>Very different specifications for the contour tolerance and the tolerance of the tool orientation can only have effect with G643.</td>
</tr>
</tbody>
</table>
Parameterization

Maximum path deviation

The maximum path deviation permitted with G642/G643 is set for each axis in the machine data:

\[ \text{MD33100 $MA\_COMPRESS\_POS\_TOL$} \]

Contour tolerance and orientation tolerance

The contour tolerance and orientation tolerance are set in the channel-specific setting data:

\[ \text{SD42465 $SC\_SMOOTH\_CONTUR\_TOL$ (maximum contour deviation)} \]
\[ \text{SD42466 $SC\_SMOOTH\_ORI\_TOL$ (maximum angular deviation of the tool orientation)} \]

The settings data can be programmed in the NC program and can in this way be specified differently for each block transition.

Note

The setting data SD42466 $SC\_SMOOTH\_ORI\_TOL$ is effective only in active orientation transformation.

Rounding behavior

Rounding behavior with G642 and G643 is configured via the machine data:

\[ \text{MD20480 $MC\_SMOOTHING\_MODE$ (rounding behavior with G64x)} \]

The units positions (E) define the behavior for G643, the tens positions (Z) the behavior for G642:

<table>
<thead>
<tr>
<th>Value E or Z</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All axes: Rounding by maintaining the maximum permitted path deviation: MD33100 $MA_COMPRESS_POS_TOL$</td>
</tr>
<tr>
<td></td>
<td>Geometry axes: Rounding by maintaining the contour tolerance: SD42465 $SC_SMOOTH_CONTUR_TOL$</td>
</tr>
<tr>
<td></td>
<td>Remaining axes: Rounding by maintaining the maximum permitted path deviation: MD33100 $MA_COMPRESS_POS_TOL$</td>
</tr>
<tr>
<td>1</td>
<td>Geometry axes: Rounding by maintaining the orientation tolerance: SD42466 $SC_SMOOTH_ORI_TOL$</td>
</tr>
<tr>
<td></td>
<td>Remaining axes: Rounding by maintaining the maximum permitted path deviation: MD33100 $MA_COMPRESS_POS_TOL$</td>
</tr>
<tr>
<td>2</td>
<td>Geometry axes: Rounding by maintaining the orientation tolerance: SD42466 $SC_SMOOTH_ORI_TOL$</td>
</tr>
<tr>
<td></td>
<td>Remaining axes: Rounding by maintaining the maximum permitted path deviation: MD33100 $MA_COMPRESS_POS_TOL$</td>
</tr>
</tbody>
</table>
### Profile for limit velocity

The use of a velocity profile for rounding in compliance with defined tolerances is controlled via the hundreds position in MD20480:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 100:</td>
<td>A profile of the limit velocity is calculated within the rounding area, based on the defined maximum values for acceleration and jerk on the participating axes or path. This can lead to an increase in the path velocity in the rounding area and therefore to the acceleration of the participating axes.</td>
</tr>
</tbody>
</table>
| ≥100: | A profile of the limit velocity is not calculated for rounding blocks with G641/G642. A constant velocity limit is specified instead. This prevents the participating axes being accelerated into the rounding area during rounding with G641/G642. However, in certain cases, this setting can cause the rounding blocks to be traversed too slowly, especially in large rounding areas.  
1xx: No velocity profile for G641  
2xx: No velocity profile for G642 |

**Note**

MD28530 $MC_MM_PATH_VELO_SEGMENTS (number of memory elements for limiting the path velocity)

### Supplementary conditions

Restriction for protection zones with active radius compensation and tool orientation:

Although tool radius compensation is applied for a tool orientation, which is not perpendicular to one of the three datum planes of the basic coordinate system, the protection zones are not rotated onto the corresponding plane.

For G643 the following must apply:

MD28530 $MC_MM_PATH_VELO_SEGMENTS > 0 (number of memory elements for limiting the path velocity)

If this condition is met, then it must be applicable for all axes:
4.3.3.3 Rounding with maximum possible axial dynamic response (G644)

Function

Maximizing the dynamic response of the axes is key to this type of continuous-path mode with rounding.

Note

Rounding with G644 is only possible if:
- All the axes involved contain only a linear motion in both the observed blocks.
- No kinematic transformation is active

In case an involved axis contains a polynomial (polynomial programmed, spline active, compressor active) or a kinematic transformation is active, the block transition is rounded with G642.

Activation

Continuous-path mode with rounding with the maximum possible axial dynamic response can be activated in any NC part program block by the modal command G644.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode with rounding with the maximum possible axial dynamic response (G644) can be deactivated by selecting:
- Modal exact stop (G60)
- Continuous-path mode G64, G641, G642, G643 or G645

Parameterization

Rounding behavior with G644 is configured via the thousands and tens of thousands places in the machine data:

MD20480 $MC_SMOOTHING_MODE (rounding behavior with G64x)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thousand's place:</td>
<td></td>
</tr>
<tr>
<td>0xxx:</td>
<td>When rounding with G644, the maximum deviations for each axis specified by the following machine data are respected:</td>
</tr>
<tr>
<td></td>
<td>MD33100 $MA_COMPRESS_POS_TOL</td>
</tr>
<tr>
<td></td>
<td>If the dynamics of the axis permit, then any specified tolerance is not utilized.</td>
</tr>
<tr>
<td>1xxx:</td>
<td>Input the maximum rounding path by programming ADIS=... or ADISPOS=...(as for G641)</td>
</tr>
</tbody>
</table>
### Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 2xxx: | Input the maximum possible frequencies of each axis in the rounding area using the machine data:  
MD32440 $MA_LOOKAH_FREQUENCY (smoothing frequency for LookAhead)  
The rounding area is defined so that no frequencies in excess of the specified maximum can occur while the rounding motion is in progress. |
| 3xxx: | Any axis that has a velocity jump at a corner traverses around the corner with the maximum possible dynamic response (maximum acceleration and maximum jerk).  
SOFT:  
If SOFT is active, the maximum acceleration and the maximum jerk of each axis is maintained.  
BRISK:  
If BRISK is active, only the maximum acceleration and not the maximum jerk of each axis is maintained.  
With this setting, neither the maximum deviations nor the rounding distance are checked.  
The resulting deviations or rounding distances are determined exclusively by the dynamic limits of the respective axis and the current path velocity. |
| 4xxx: | As in case of 0xxx, the maximum deviations of each axis specified with the following machine data are used:  
MD33100 $MA_COMPRESS_POS_TOL  
Contrary to 0xxx, the specified tolerance is also utilized, if possible. Therefore, the axis does not attain its maximum possible dynamics. |
| 5xxx: | As in case of 1xxx, the maximum possible rounding path is specified through programming of ADIS=... or ADISPOS= respectively.  
Contrary to 1xxx, the specified rounding path is also utilized, if possible. Therefore, the axes involved do not attain their maximum possible dynamics. |

### Ten thousands digit

| 0xxxx | The velocity profiles of the axes in the rounding area are determined without jerk limiting for BRISK and with jerk limiting for SOFT. |
| 1xxxx | The velocity profiles of the axes in the rounding area are always determined with jerk limiting, regardless of whether BRISK or SOFT is active. |

When specifying the maximum axial deviations (MD33100 $MA_COMPRESS_POS_TOL) or the maximum rounding distance (ADIS / ADISPOS) the available rounding path is normally not used, if permitted by the dynamics of the axes involved. Through this, the length of the rounding path depends on the active path feedrate. In case of lower path speeds, one gets lower deviations from the programmed contours. However, it can be set that in these cases the specified maximum axial deviation or the specified rounding distance is utilized, if possible. In this case the deviations depend on the programmed contour independent of the programmed path feedrate.

### Note

Apart from the ones mentioned, the following limitation can also become active additionally:

The rounding distance cannot exceed half the length of the original participating blocks.
Jerk limitation

The smoothing of the velocity jump on each axis and thus the shape of the rounding path depends on whether an interpolation is performed with or without jerk limitation. Without jerk limitation the acceleration of each axis reaches its maximum value in the entire rounding area.
With jerk limitation, the jerk of each axis is limited to its maximum value within the rounding area. The rounding motion thus generally consists of three phases:

- **Phase 1**
  During phase 1, each axis builds up its maximum acceleration. The jerk is constant and equal to the maximum possible jerk on the respective axis.

- **Phase 2**
  During phase 2, the maximum permissible acceleration is applied.

- **Phase 3**
  During phase 3, which is the last phase, the acceleration of each axis is reduced back to zero with the maximum permissible jerk.

### 4.3.3.4 Rounding of tangential block transitions (G645)

**Function**

In continuous-path mode with rounding, rounding blocks are also only generated on tangential block transitions if the curvature of the original contour exhibits a jump in at least one axis.

The rounding motion is defined here so that the acceleration of all axes involved remains smooth (no jumps) and the parameterized maximum deviations from the original contour (MD33120 $MA\_PATH\_TRANS\_POS\_TOL$) are not exceeded.

In the case of angular, non-tangential block transitions, the rounding behavior is the same as with G642 (see Section "Rounding in compliance with defined tolerances (G642/G643) (Page 192)". 

![Diagram of rounding motion with phases]
Activation/deactivation

Continuous-path mode with rounding of tangential block transitions can be activated in any NC part program block by the modal command G645.

Selecting the exact stop which works on a block-by-block basis enables rounding to be interrupted (G9).

Continuous-path mode with rounding of tangential block transitions (G645) can be deactivated by selecting:

- Modal exact stop (G60)
- Continuous-path mode G64, G641, G642, G643 or G644

Comparison between G642 and G645

When rounding with G642, the only block transitions rounded are those which form a corner, i.e. the velocity of at least one axis jumps. However, if a block transition is tangential, but there is a jump in the curvature, no rounding block is inserted with G642. If this block transition is traversed with finite velocity, the axes experience some degree of jump in acceleration which (with the jerk limit activated!) may not exceed the parameterized limit (MD32432 $MA\_PATH\_TRANS\_JERK\_LIM). Depending on the level of the limit, the path velocity at the block transition may be greatly reduced as a result. This constraint is avoided by using G645 because the rounding motion is defined here in such a way that no jumps occur in acceleration.

Parameterization

The following machine data indicates the maximum permissible path deviation for each axis during rounding with G645:

MD33120 $MA\_PATH\_TRANS\_POS\_TOL

This value is only of relevance to tangential block transitions with variable acceleration. When angular, non-tangential block transitions are rounded, (as with G642) the tolerance from MD33100 $MA\_COMPRESS\_POS\_TOL becomes effective.

See also

Free-form surface mode: Basic functions (Page 224)

4.3.3.5 Rounding and repositioning (REPOS)

If the machining in the area of the rounding contour is interrupted, a REPOS operation cannot be used to position again directly on the rounding contour. In this case, positioning can be made only on the programmed contour.

Example

Programmed: Two traversing blocks N10 and N20 with programmed rounding G641.

The traversing motion is interrupted in the rounding area. The axes, e.g. manually, are then traversed to the REPOS start point. Depending on the selected REPOS mode, the repositioning on the contour is made at the points ①, ② or ③.
4.3.4 LookAhead

4.3.4.1 Standard functionality

Function

LookAhead is a function which is active in continuous-path mode (G64, G64x) and determines a foreseeable velocity control for multiple NC part program blocks over and beyond the current block.

Note

LookAhead is only available for path axes, not for spindles and positioning axes.

If a part program contains consecutive blocks with very small paths, only one velocity is reached per block without LookAhead, enabling deceleration of the axes at the end of the block while maintaining acceleration limits. This means that the programmed velocity is not reached at all. With LookAhead, however, it is possible to implement the acceleration and deceleration phase over multiple blocks with approximately tangential block transitions, thereby achieving a higher feedrate with shorter distances.
Deceleration to velocity limits is possible with LookAhead such that violation of the acceleration and velocity limit is prevented.

LookAhead takes plannable velocity limits into consideration such as:

- Exact stop at block end
- Velocity limit in the block
- Acceleration limit in the block
- Velocity limit on block transition
- Synchronization with block change at block transition.

Mode of operation

LookAhead carries out a block-specific analysis of velocity limits and specifies the required brake ramp profile based on this information. LookAhead is adapted automatically to block length, braking capacity and permissible path velocity.

For safety reasons, the velocity at the end of the last prepared block must initially be assumed to be zero because the next block might be very small or be an exact-stop block, and the axes must have been stopped by the end of the block.

With a series of blocks with high set velocity and very short paths, the speed can be increased in each block depending on the velocity value currently calculated by the LookAhead function in order to achieve the required set velocity. After this it can be reduced so that the velocity at the end of the last block considered by the LookAhead function can be zero. This results in a serrated velocity profile (see the following fig.) which can be avoided by reducing the set velocity or increasing the number of blocks considered by the LookAhead function.
Activation/deactivation

LookAhead is activated by selecting continuous-path mode G64, G641, G642, G643, G644 or G645.

Selecting the exact stop which works on a non-modal basis enables rounding to be interrupted (G09).

LookAhead is deactivated by selecting the modal exact stop (G60).

Parameterization

Number of blocks

To achieve reliable axis traversal in continuous-path mode, the feedrate must be adapted over several blocks. The number of blocks considered by the LookAhead function is calculated automatically and can, if required, be limited by a machine data. The default setting is "1", which means that LookAhead only considers the following block for velocity control.

Because LookAhead is especially important for short blocks (relative to the deceleration path), the number of blocks required is of interest for LookAhead braking. It is sufficient to consider the path length to be equal to the deceleration path that is required to brake from maximum velocity to standstill.

For a machine with a low axial acceleration of \(a = 1 \text{ m/s}^2\) and a high feedrate of \(v_{\text{path}} = 10 \text{ m/min}\), the following number of \(n_{\text{LookAhead}}\) blocks are allocated to the controller where it has has an attainable block cycle time of \(T_B = 10 \text{ ms}\):

\[
 n_{\text{LookAhead}} = \frac{\text{Deceleration path/Block length}}{v_{\text{path}}^2 / (2a)} \cdot (v_{\text{path}} \cdot T_B) = 9
\]

Considering these conditions, it is advisable to adapt the feedrate over 10 blocks. The number of blocks entered for the LookAhead function forecast does not change the LookAhead algorithm and memory requirement.

Since the machining velocity is very often set to a lower value than the maximum velocity in a program, more blocks than are required would be predicted, overloading the processor.

Figure 4-6 Example for modal velocity control (number of blocks considered by the LookAhead function = 2)
unnecessarily. For this reason, the required number of blocks is derived from the velocity which is calculated from the following multiplication:

- Programmed velocity * MD12100 $MN_OVR_FACTOR_LIMIT_BIN
  (when using a binary-coded feedrate override switch)
- Programmed velocity * MD12030 $MN_OVR_FACTOR_FEEDRATE[30]
  (when using a gray-coded feedrate override switch)

The value for MD12100 or the 31st override value for MD12030 defines the dynamic response reserves which the velocity control provides for when the path feed is overshot.

**Note**
The 31st override value for MD12030 should correspond to the highest override factor which is actually used.

**Note**
The number of blocks considered by the LookAhead function is limited by the possible number of NC blocks in the IPO buffer.

**Velocity profiles**
In addition to the fixed, plannable velocity limitations, LookAhead can also take account of the programmed velocity. This makes it possible to achieve a lower velocity by applying LookAhead beyond the current block.

- **Determination of the following block velocity**
  One possible velocity profile contains the determination of the following block velocity.
  Using information from the current and the following NC block, a velocity profile is calculated from which, in turn, the required velocity reduction for the current override is derived.
  The calculated maximum value of the velocity profile is limited by the maximum path velocity. With this function it is possible to initiate a speed reduction in the current block taking override into account such that the lower velocity of the following block can be achieved. If the reduction in velocity takes longer than the travel time of the current block, the velocity is further reduced in the following block. Velocity control is only ever considered for the following block.
  The function is activated via the machine data:
  MD20400 $MC_LOOKAH_USE_VELO_NEXT_BLOCK

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Function active</td>
</tr>
<tr>
<td>FALSE</td>
<td>Function not active</td>
</tr>
</tbody>
</table>
• **Definition of override points**

If the velocity profile of the following block velocity is not sufficient because, for example, very high override values (e.g., 200%) are used or a constant cutting rate G96/G961 is active, with the result that the velocity must be further reduced in the following block, LookAhead provides a way of reducing the programmed velocity over several NC blocks. By defining override points, LookAhead calculates a limiting velocity profile for each value. The required velocity reductions for the current override are derived from these profiles. The calculated maximum value of the velocity profile is limited by the maximum path velocity. The upper point should cover the velocity range that will be reached by the maximum value set in the machine data:

MD12030 $MN_OVR_FACTOR_FEEDRATE [n] (evaluation of the path feedrate override switch)

It can also be reached via the value of the machine data:

MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary coded override switch)

In this way, a reduction of the velocity continuing into the block in which it is programmed can be avoided.

If velocity reductions across block boundaries are required at a 100% override, a point should be set in the lower override range as well.

The number of override points used per channel is specified in the machine data:

MD20430 $MC_LOOKAH_NUM_OVR_POINTS (number of override switch points for LookAhead)

The associated points are stored in the machine data:

MD20440 $MC_LOOKAH_OVR_POINTS (prepared override velocity characteristics with LookAhead)

Example:

Limiting velocity characteristics, whereby:

- Override = 50%, 100% or 150%
- Number of LookAhead blocks = 4
- MD20430 $MC_LOOKAH_NUM_OVR_POINTS = 2
- MD20440 $MC_LOOKAH_OVR_POINTS = 1.5, 0.5
A combination of both procedures (determination of following block velocity and determination of override points) can be used to calculate the velocity profiles and is generally advisable because the preset machine data for these functions already takes the widest range of override-dependent velocity limits into account.

**Note**

If neither of the procedures has been activated, the setpoint velocity is always applied in the current block.

**Note**

Plannable velocity limits restrict override-specific velocity limits.

### Relief factor with block cycle problems

Block cycle problems are encountered in cases where the traversing distances of the NC blocks to be processed are so short that the LookAhead function has to reduce the machine velocity to provide enough time for block processing. In this situation, constant braking and acceleration of path motion may occur.

Velocity fluctuations of this type can be dampened by specifying a relief factor:

MD20450 $MC_LOOKAH_RELIEVE_BLOCK_CYCLE (relieving factor for the block cycle time)
Supplementary conditions

**Axis-specific feed stop/axis disable**

Axis-specific feed stop and axis-specific axis disable are ignored by LookAhead.

If an axis is to be interpolated that should on the other hand be made stationary by axis-specific feed stop or axis disable, LookAhead does not stop path motion before the block in question but decelerates in the block itself.

If this response is not wanted, an axis-specific feed stop can be transferred to a channel via the PLC to stop the path immediately (see also Section "Clamping monitoring (Page 86)").

4.3.4.2 Free-form surface mode: Extension function

**Function**

The "Free-form surface mode: Extension function" is an extension of the Look Ahead standard functionality and is used to calculate the path velocity profile during free-form surface processing (see also Section "Free-form surface mode: Basic functions (Page 224)").

Its use optimizes the continuous-path mode as follows:

- Symmetry between the acceleration and deceleration profiles
- Uniform acceleration process, even with changing jerk or acceleration limits
- Uniform acceleration process of target velocity profiles, irrespective of the degree to which they can or cannot be started with the specified dynamic response limit
- Look Ahead braking to lower setpoint velocities

Uniformity and compliance with the dynamic response limit guarantee that the setpoint velocity profiles are smoothed to a homogeneous velocity profile on the part. This serves to minimize the effect of following errors on the quality of the surface.

**Advantages**

- Greater uniformity in the surface of the workpiece
- Lower machine load

**Applications**

The "Free-form surface mode: Extension function" is used to process workpieces which primarily comprise free-form surfaces.

**Note**

As better results are not achieved for standard processing applications, standard Look Ahead functionality should be used in these cases.
Activation

The function is only effective:

- In AUTOMATIC
- In the "Acceleration with jerk limit (SOFT)" mode

Parameterization

Working memory

The memory for the "Free-form surface mode: Extended function" is configured via the machine data:

$$\text{MD28533 } \text{MC\_MM\_LOOKAH\_FFORM\_UNITS} = \text{<value>}\)$$

The required memory depends on the part program, the block lengths, the axis dynamic response, as well as on an active kinematic transformation.

The following setting applies as a guideline for processing free-form surfaces: MD28533 = 18

Note

Due to the additional storage requirements, MD28533 should only be set for the channels in which free-form surfaces are being processed.

Number of NC blocks in the IPO buffer

It is generally advisable to significantly increase the configured number of NC blocks in the interpolation buffer when using the "Free-form surface mode: Extension function":

$$\text{MD28060 } \text{MC\_MM\_IPO\_BUFFER\_SIZE} > 100$$

If the block memory capacity is too low, this may diminish the uniformity of the path-velocity profile.

Activation/deactivation

The function can be switched on or off independently for every dynamic response mode (see Section "Dynamic response mode for path interpolation (Page 222)"):

$$\text{MD20443 } \text{MC\_LOOKAH\_FFORM[<n>]} = \text{<value>}$$

<table>
<thead>
<tr>
<th>Index &lt;n&gt;</th>
<th>Dynamic response mode</th>
<th>&lt;value&gt;</th>
<th>Free-form surface mode: Extension function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Standard dynamic response settings (DYNNORM)</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>On</td>
</tr>
<tr>
<td>1</td>
<td>Positioning mode, tapping (DYNPOS)</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>On</td>
</tr>
<tr>
<td>2</td>
<td>Roughing (DYNROUGH)</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>On</td>
</tr>
<tr>
<td>3</td>
<td>Finishing (DYNSEMIFIN)</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>On</td>
</tr>
</tbody>
</table>
4.3 Continuous-path mode

<table>
<thead>
<tr>
<th>Index &lt;n&gt;</th>
<th>Dynamic response mode</th>
<th>&lt;value&gt;</th>
<th>Free-form surface mode: Extension function</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Smooth finishing (DYNFINISH)</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>On</td>
</tr>
</tbody>
</table>

The "Free-form surface mode: Extension function" is typically only active if the "Free-form surface mode: Basic functions" are also active. Therefore, the settings in MD20443 $MC_LOOKAH_FFORM[<n>] should correspond to the settings in MD20606 $MC_PREPDYN_SMOOTHING_ON[<n>].

The standard Look Ahead functionality is active in the dynamic response modes in which the "Free-form surface mode: Extension function" is switched off.

**Programming**

Generally speaking, the "Free-form surface mode: Extension function" becomes effective as a result of a change in the dynamic response mode in the part program.

**Example**

The following parameters are assumed:

- MD20443 $MC_LOOKAH_FFORM[0] = 0
- MD20443 $MC_LOOKAH_FFORM[1] = 0
- MD20443 $MC_LOOKAH_FFORM[2] = 1
- MD20443 $MC_LOOKAH_FFORM[3] = 1
- MD20443 $MC_LOOKAH_FFORM[4] = 1

Change of the dynamic response mode in the part program:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 DYNPOS</td>
<td>; Activate DYNPOS dynamic response mode. Standard Look Ahead functionality is active in the DYNPOS dynamic response mode.</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>N100 G17 G54 F10000</td>
<td></td>
</tr>
<tr>
<td>N101 DYNFINISH</td>
<td>; Activate DYNFINISH dynamic response mode. The &quot;Free-form surface mode: Extension functions&quot; are active in the DYNFINISH dynamic response mode.</td>
</tr>
<tr>
<td>N102 SOFT G642</td>
<td></td>
</tr>
<tr>
<td>N103 X-0.274 Y149.679 Z100.000 G0</td>
<td></td>
</tr>
<tr>
<td>N104 COMPCAD</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N1009 Z4.994 G01</td>
<td></td>
</tr>
<tr>
<td>N10010 X.520 Y149.679 Z5.000</td>
<td></td>
</tr>
<tr>
<td>N10011 X10.841 Y149.679 Z5.000</td>
<td></td>
</tr>
</tbody>
</table>
Program code

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10012 X11.635 Y149.679 Z5.010</td>
<td></td>
</tr>
<tr>
<td>N10013 X12.032 Y149.679 Z5.031</td>
<td></td>
</tr>
<tr>
<td>M30</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

When switching between the standard Look Ahead functionality and the "Free-form surface mode: Extension function" or vice versa, continuous-path mode is interrupted by an interpolator stop.

**Supplementary conditions**

**Automatic function switchover**

Use of the following functions results in an automatic switchover to standard Look Ahead functionality:

- Thread cutting/tapping (G33, G34, G35, G331, G332, G63)
- Path master-value coupling
- Punching, nibbling
- Cartesian PTP travel

The "Free-form surface mode: Extension function" is then switched on again automatically.

**Using the commands of G function group 15 (feed types)**

The following feed types are not recommended in conjunction with the "Free-form surface mode: Extension function".

- Feedrate per revolution (G95, G96, G97, etc.)
- Inverse-time feedrate (G93)

**Use of FLIN**

The FLIN feedrate profile cannot be used in conjunction with the "Free-form surface mode: Extension function".

**Influence of feedrate overrides**

Feedrate overrides (via a machine control panel, $AC_OVR, ...) can extend the traverse time over standard Look Ahead functionality considerably.

**Interaction with rapid traverse motion (G0)**

G0 blocks which are interspersed during free-form surface processing do not switch the Look Ahead functionality over (from the "Free-form surface mode: Extended function" to the standard Look Ahead functionality or vice versa). This means that even though the standard dynamic response setting (DYNNORM) is effective with G0, the standard Look Ahead functionality which is preset for DYNNORM (→ MD20443 $MC_LOOKAH_FFORM[0]) does not automatically become effective as well. By retaining the Look Ahead functionality which is currently active, a more homogeneous velocity profile is achieved, particularly since G0 and polynomial blocks are usually smoothed and connected by rounding.
4.4 Dynamic adaptations

4.4.1 Smoothing of the path velocity

Introduction

The velocity control function utilizes the specified axial dynamic response. If the programmed feedrate cannot be achieved, the path velocity is brought to the parameterized axial limit values and the limit values of the path (velocity, acceleration, jerk). This can lead to repeated braking and acceleration on the path.

If a short acceleration takes place during a machining function with high path velocity, and is thus followed almost immediately by braking, the reduction in the machining time is only minimal. Acceleration of this kind can, however, have undesirable effects if, for example, it results in machine resonance.

In some applications in mold making, especially in the case of high-speed cutting, it is desirable to achieve a constant path velocity. In these cases, it can therefore be reasonable to sacrifice transient acceleration processes in favor of a smoother tool path velocity.

Function

If the "smoothing the path velocity" function is active, a smoothing factor, which determines the maximum permissible productivity loss, takes effect with a view to achieving smoother path velocity control: Acceleration processes which contribute less than this factor to a shorter program runtime are not performed. Account is only taken of acceleration processes whose frequencies lie above the configurable limit frequencies of the axes involved.

Benefits:

- Avoidance of excitations of possible machine resonance due to continuous, transient braking and acceleration processes (in the area of less IPO cycles).
- Avoidance of constantly varying cutting rates due to acceleration which brings no significant shortening of the program running time.

Note

The smoothing of the path velocity does not lead to contour errors.

Variations in axis velocity due to curvatures in the contour at constant path velocity may continue to occur and are not reduced with this function.

Variations in path velocity due to the input of a new feedrate are not changed either. This remains the responsibility of the programmer of the subprogram.
Requirements

- The smoothing of the path velocity is only effective in continuous-path mode with LookAhead over multiple blocks with SOFT and BRISK. Smoothing is not effective with G0.
- The controller's cycle times must be configured in such a way that preprocessing can prepare sufficient blocks to enable an acceleration process to be analyzed.

Activation/deactivation

The "smoothing of the path velocity" function is activated/deactivated with the machine data:
MD20460 $MC_LOOKAH_SMOOTH_FACTOR (smoothing factor for LookAhead)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>Smoothing of the path velocity not active</td>
</tr>
<tr>
<td></td>
<td>(default)</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>Smoothing of the path velocity active</td>
</tr>
</tbody>
</table>

A change in the MD setting is only made effective through NEW CONF.

Parameterization

Smoothing factor

The smoothing factor is set via the channel-specific machine data:
MD20460 $MC_LOOKAH_SMOOTH_FACTOR (smoothing factor for LookAhead)

The percentage value defines how much longer a processing step without accelerations/decelerations may be than the corresponding step with accelerations/decelerations.

This would be a "worst-case" value, if all accelerations within the part program, except the initial approach motion, were smoothed. The actual extension will always be smaller, and may even be 0, if the criterion is not met by any of the accelerations. Values between 50 and 100% may also be entered without significantly increasing the machining time.

Consideration of the programmed feed

The path velocity can be smoothed with or without taking the programmed feedrate into consideration. The selection is made via the machine data:
MD20462 $MC_LOOKAH_SMOOTH_WITH_FEED (path smoothing with programmed feedrate)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Programmed feedrate is not taken into</td>
</tr>
<tr>
<td></td>
<td>consideration.</td>
</tr>
<tr>
<td>1</td>
<td>Programmed feedrate is considered (default</td>
</tr>
<tr>
<td></td>
<td>setting).</td>
</tr>
</tbody>
</table>

When considering the programmed feedrate, the specified smoothing factor (see MD20460) is maintained better when the override is 100%.

Axis-specific limit frequencies

The axis-specific limit frequencies are defined via the machine data:
MD32440 $MA_LOOKAH_FREQUENCY (smoothing frequency for LookAhead)
Acceleration and deceleration processes, which run with a high frequency, are smoothed depending upon the parameterization of the following machine data or else are reduced in dynamics:

MD20460 $MC_LOOKAH_SMOOTH_FACTOR (smoothing factor for LookAhead)
MD20465 $MC_ADAPT_PATH_DYNAMIC (adaptation of the dynamic path response)

For further information on MD20465, see Section "Adaptation of the dynamic path response (Page 213)">

**Note**

If vibrations are generated in the mechanical system of an axis and if the corresponding frequency is known, MD32440 should be set to a value smaller than this frequency.

The needed resonance frequencies can be calculated using the built-in measuring functions.

### Mode of operation

The minimum value for MD32440 is calculated as $f_{path}$ on the basis of the axes involved in the path. For the smoothing only those acceleration processes are taken into consideration, in which the start and the end velocity of this motion are reached within the time given below:

$$ t = t_2 - t_1 = \frac{2}{f_{path}} $$

These acceleration processes are dispensed with if the resulting extension in the processing time does not exceed the limit specified in excess of the smoothing factor (MD20460).

### Example

The following parameters are assumed:

MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 10$

MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 20 Hz$

MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 20 Hz$

MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 10 Hz$

The path involves the three axes X = AX1, Y = AX2, Z = AX3.

The minimum value of MD32440 for these three axes is thus 10 Hz. This means that any acceleration, which is completed within a period of $t_2 - t_1 = 2/10 \text{ Hz} = 200 \text{ ms}$, is examined. The time $t_2$ is the time it takes to reach velocity $v_1$ again following an acceleration process starting from velocity $v_1$. The extending of the execution time is also only considered within this range.

If the time $t_2 - t_1$ is greater than 200 ms or if the additional program execution time $t_3 - t_2$ is more than 10% (= MD20460) of $t_2 - t_1$, the following time characteristic applies:
Figure 4-7  Characteristic of time-optimum path velocity (without smoothing)

If, however, the time $t_2 - t_1$ is less than 200 ms or if the additional program execution time $t_3 - t_2$ is no more than 10% of $t_2 - t_1$, the following time characteristic applies:

Figure 4-8  Characteristic of the smoothed path velocity

### 4.4.2 Adaptation of the dynamic path response

**Function**

Highly dynamic acceleration and deceleration processes during machining can cause excitation of mechanical vibrations of machine elements and consequently a reduction of the surface quality of the workpiece.
The dynamic response of the acceleration and deceleration processes can be adapted to the machine conditions using the "adaptation of the dynamic path response" function.

**Note**

The "adaptation of the dynamic path response" function only concerns the resulting path and not the deceleration and acceleration processes of the individual axes involved in the path. For this reason, critical deceleration and acceleration processes of the axes with respect to the excitation of mechanical vibrations can occur due to discontinuous contour profiles or kinematic transformations, even with a constant path velocity profile.

**Effectiveness**

The "adaptation of the dynamic path response" function is only effective during path motions:

- Continuous-path mode (G64, G64x)
  
  In continuous-path mode, the optimal effect of the dynamic response adaptation is attained with an active 100% override. Considerable deviations from this value or functions that cause the path axes to decelerate (e.g. auxiliary function outputs to the PLC) greatly reduce the desired action.

- Exact stop (G60)

In addition, the "adaptation of the dynamic path response" function is **not** active during path motions:

- Programmed rapid traverse (G0)
- Changes in the override value
- Stop requests during motion (e.g. NC Stop, NC Reset)
- "Velocity-dependent path acceleration" function (DRIVE) is active

**Activation/deactivation**

The function is activated/deactivated with the machine data:

MD20465 $MC_ADAPT_PATH_DYNAMIC (adaptation of the dynamic path response)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\leq 1.0</td>
<td>Dynamic adaptation <strong>not</strong> active (default setting)</td>
</tr>
<tr>
<td>&gt; 1.0</td>
<td>Dynamic adaptation active</td>
</tr>
</tbody>
</table>

When activation takes place, the "**smoothing the path velocity**" function is always activated internally in continuous-path mode as well (see Section "Smoothing of the path velocity (Page 210)").

If the smoothing factor (MD20460 $MC_LOOKAH_SMOOTH_FACTOR) is set to 0% (= function deactivated; default!), a smoothing factor of 100% is used as a substitute. For a smoothing factor other than 0%, the set value takes effect.
Parameterization

**Adaptation factor of the dynamic path response**

Via the adaptation factor of the dynamic path response, temporary changes in the path velocity are executed with smaller dynamic response limit values.

The adaptation factor is to be set on a channel-specific basis:

- For traversing motions with acceleration without jerk limitation (BRISK) via: MD20465 $MC_ADAPT_PATH_DYNAMIC[ 0 ]
  → The adaptation factor acts on the acceleration.

- For traversing motions with acceleration with jerk limitation (SOFT) via: MD20465 $MC_ADAPT_PATH_DYNAMIC[ 1 ]
  → The adaptation factor acts on the jerk.

**Axis-specific limit frequencies**

The dynamic response limiting should only be active during deceleration and acceleration processes that trigger mechanical vibrations larger than a specific limiting frequency, thus causing excitation of machine resonances.

This limit frequency from which the dynamic response limiting activates, is specified on an axis-specific basis via the machine data:

MD32440 $MA_LOOKAH_FREQUENCY (smoothing frequency for LookAhead)

For further information, see Section "Smoothing of the path velocity (Page 210)".

**Mode of operation**

During processing and via all the axes involved in the path, the controller cyclically establishes the minimum of all the limit frequencies to be the limit frequency \( f \) for the adaptation of the dynamic response and calculates the relevant time window \( t_{\text{adapt}} \) from this:

\[ t_{\text{adapt}} = \frac{1}{f} \]

The size of the relevant time window \( t_{\text{adapt}} \) determines the further behavior:

1. The time needed to change the velocity is less than \( t_{\text{adapt}} \):
   - The acceleration rates are reduced by a factor \( > 1 \) and \( \leq \) the value written in machine data: MD20465 ADAPT_PATH_DYNAMIC (adaptation of the path dynamics)
   - The reduction in acceleration rate increases the time taken to change the velocity.
   - The following cases are different:
     - The acceleration rate is reduced with a value less than MD20465 so that the process lasts for \( t_{\text{adapt}} \). The permitted reduction does not need to be fully utilized.
     - The acceleration time is reduced with the value written in MD20465. The process lasts less than \( t_{\text{adapt}} \) despite the reduced acceleration. The permitted reduction was fully utilized.

2. The time needed to change the velocity is greater than \( t_{\text{adapt}} \):
   - No dynamic response adaptation is required.
Example

The following example is intended to show the effect of the "adaptation of the dynamic path response" function on traversing motions with acceleration and without jerk limitation (BRISK).

The following parameters are assumed:

- $MC\_ADAPT\_PATH\_DYNAMIC[0] = 1.5$
- $MC\_LOOKAH\_SMOOTH\_FACTOR = 1.0$
- $MA\_LOOKAH\_FREQUENCY[AX1] = 20$ Hz  \( T_{AX1} = 1/20 \text{ Hz} = 50 \text{ ms} \)
- $MA\_LOOKAH\_FREQUENCY[AX2] = 10$ Hz  \( T_{AX2} = 1/10 \text{ Hz} = 100 \text{ ms} \)
- $MA\_LOOKAH\_FREQUENCY[AX3] = 20$ Hz  \( T_{AX3} = 1/20 \text{ Hz} = 50 \text{ ms} \)

Note

To illustrate the effect of dynamic response adaptation, the value for the smoothing factor (MD20460) is set to "1", whereby the "smoothing of the path velocity" function is practically deactivated.

The path involves the three axes $X = AX1$, $Y = AX2$, $Z = AX3$.

For path motions in which axis AX2 is involved, all deceleration and acceleration processes that would last less than $T_{AX2}$ are adapted.

If only axes AX1 and/or AX3 are involved in path motions, all deceleration and acceleration processes that would last less than $T_{AX1} = T_{AX3}$ are adapted.

The relevant time window is marked $t_{adapt}$ in the figures below.

![Figure 4-9](image)

Figure 4-9  Path velocity profile optimized for time without smoothing or dynamic adaptation response
Figure 4-10  Path velocity profile with adaptation of dynamic path response

Intervals $t_0 - t_1$ and $t_2 - t_3$: The acceleration process between $t_0 - t_1$ and the deceleration process between $t_2 - t_3$ are extended in terms of time to $t_{\text{adapt01}}$ or $t_{\text{adapt23}}$ as a result of the acceleration being adapted.

Interval $t_4 - t_5$: The acceleration process between $t_4 - t_5$ is executed with an acceleration reduced by the maximum adaptation factor of 1.5. However, the acceleration process is completed before time $t_{\text{adapt45}}$.

Interval $t_6 - t_7$: The deceleration process between $t_6 - t_7$ remains unchanged as it lasts longer than $t_{\text{adapt67}}$.

4.4.3 Determination of the dynamic response limiting values

In addition to determining the natural frequency of the path axes for assigning parameters to the axis-specific limit frequencies (MD32440 $\$\text{MA\_LOOKAH\_FREQUENCY}$), the implementation of the "adaptation of the dynamic path response" function also requires dynamic response limits to be determined for velocity, acceleration and jerk.
4.4 Dynamic adaptations

Procedure

The determination of the dynamic response limits for the traversing of path axes by means of acceleration with jerk limiting (SOFT) is described below. This procedure can be applied by analogy to the case of acceleration without jerk limiting (BRISK).

1. Deactivate the "adaptation of the dynamic path response" function:
   
   MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = 1

2. Observe the positioning behavior of each path axis at different traversing velocities. When doing so, set the jerk such that the desired positioning tolerance is maintained.

   Note
   The higher the traversing velocity from which the positioning process is started, the higher in general the jerk can be set.

3. Use the maximum permissible jerk determined for the least critical traversing velocity:
   
   MD32431 $MA_MAX_AX_JERK (maximum jerk)

4. Determine the $F_{APD}$ factor for all of the path axes using:
   
   $F_{APD} = \frac{\text{largest determined jerk}}{\text{smallest determined jerk}}$

   Note
   The smallest determined jerk is the value for the jerk during the most critical traversing velocity.

5. Enter the largest $F_{APD}$ factor that was determined via all the path axes as the value for the adaptation factor for the path dynamic response:
   
   MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = F_{APD}$

4.4.4 Interaction between the "smoothing of the path velocity" and "adaptation of the path dynamic response" functions

The following examples serve to illustrate the interaction between the "smoothing of the path velocity" and "adaptation of the path dynamic response" functions in continuous-path mode.

Example 1

Acceleration mode: BRISK

The path involves the 3 axes X = AX1, Y = AX2, Z = AX3.

The following parameters are assumed:

MD20465 $MC_ADAPT_PATH_DYNAMIC[0] = 3
MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 80.0
MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 20 T_{AX1} = 1/20 Hz = 50 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 20 T_{AX2} = 1/20 Hz = 50 ms
MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 20 T_{AX3} = 1/20 Hz = 50 ms
Effects of smoothing on path velocity:

Interval $t_1 - t_2$: The acceleration and deceleration process between $t_1$ and $t_2$ does not take place because the lengthening of the machining time without the acceleration process to $v_{12}$ is less than the resulting time if a smoothing factor of 80 % is applied.

Interval $t_3 - t_5$: The acceleration and braking profile between $t_3$ and $t_5$ does not fulfill this condition or takes longer than the parameterized smoothing time $T_{Axn} = 2/20 \text{ Hz} = 100 \text{ ms}$.

Effects of the dynamic response adaptation:

Interval $t_3 - t_4$: The acceleration process between $t_3$ and $t_4$ is shorter than $\text{MIN}(T_{Axn}) = 1/20 \text{ Hz} = 50 \text{ ms}$ and is, therefore, executed with an acceleration reduced by an adaptation factor of 3.

Interval up to $t_1$: The acceleration up to $t_1$ left over after path smoothing is stretched to the time period up to $t_1'$ by the dynamic response adaptation.
Note
The example shows that those acceleration or deceleration processes that are not eliminated by the smoothing of the path velocity can be subsequently optimized by adapting the dynamic path response. For this reason, both functions should always be activated, if possible.

Example 2
Acceleration mode: SOFT
The path involves the 3 axes X = AX1, Y = AX2, Z = AX3.
The following parameters are assumed:

- MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = 1
- MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 0.0
- MD32440 $MA_LOOKAH_FREQUENCY[AX1] = 10 \quad T_{AX1} = 1/20 \text{ Hz} = 100 \text{ ms}
- MD32440 $MA_LOOKAH_FREQUENCY[AX2] = 10 \quad T_{AX2} = 1/20 \text{ Hz} = 100 \text{ ms}
- MD32440 $MA_LOOKAH_FREQUENCY[AX3] = 20 \quad T_{AX3} = 1/20 \text{ Hz} = 50 \text{ ms}

This leads to a path velocity profile which is optimized in terms of time without smoothing the path velocity or adapting the dynamic path response:

The parameter assignment is changed as follows:

- MD20465 $MC_ADAPT_PATH_DYNAMIC[1] = 4
- MD20460 $MC_LOOKAH_SMOOTH_FACTOR = 1.0

This results in a path velocity profile with adaptation of the dynamic path response and with minimum, and thus virtually deactivated, smoothing of the path velocity:
The smoothing factor is set to 0% instead of 1% (in accordance with the default!):

\[
\text{MD20460 } \text{SMC\_LOOKAH\_SMOOTH\_FACTOR} = 0.0
\]

A smoothing factor of 100% comes into effect with this parameter assignment.

This gives rise to a path velocity profile with smoothing of the path velocity and adaptation of dynamic path response:

---

**B1: Continuous-path mode, Exact stop, Look Ahead**

4.4 Dynamic adaptations
4.4.5 Dynamic response mode for path interpolation

Function

Technology-specific, dynamic response settings can be saved in machine data and can be activated in the part program via the commands from G function group 59 (dynamic response mode for path interpolation).

<table>
<thead>
<tr>
<th>Command</th>
<th>Activates the dynamic response settings for:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DYNNORM</td>
<td>Standard dynamic response settings</td>
</tr>
<tr>
<td>DYNPOS</td>
<td>Positioning mode, tapping</td>
</tr>
<tr>
<td>DYNROUGH</td>
<td>Roughing</td>
</tr>
<tr>
<td>DYNSEMIFIN</td>
<td>Finishing</td>
</tr>
<tr>
<td>DYNFINISH</td>
<td>Smooth finishing</td>
</tr>
</tbody>
</table>

Note

The dynamic response of the path axes alone is determined by the commands from G function group 59 (dynamic response mode for path interpolation). They have no effect on:

- Positioning axes
- PLC axes
- Command axes
- Motions based on axis coupling
- Overlaid motions with handwheel
- JOG motions
- Reference point approach (G74)
- Fixed-point approach (G75)
- Rapid traverse motion (G0)

The standard dynamic response setting (DYNNORM) always takes effect for these axis motions.

Application

By switching the dynamic response settings, roughing can be optimized in terms of time and smoothing can be optimized in terms of the surface, for example.

Parameterization

Axis-specific dynamic response settings:

- MD32300 $MA_MAX_AX_ACCEL\[<n>\]$ (axis acceleration)
- MD32310 $MA_MAX_ACCEL_OVL_FACTOR\[<n>\]$ (overload factor for axial jumps in velocity)
- MD32431 $MA_MAX_AX_JERK\[<n>\]$ (maximum axial jerk for path motion)
• MD32432 $MA_PATH_TRANS_JERK_LIM[n>] (maximum axial jerk at the block transition in continuous-path mode)

• MD32433 $MA_SOFT_ACCEL_FACTOR[n>] (scaling of the acceleration limitation with SOFT)

**Channel-specific dynamic response settings:**

• MD20600 $MC_MAX_PATH_JERK[n>] (path-related maximum jerk)

• MD20602 $MC_CURV_EFFECT_ON_PATH_ACCEL[n>] (influence of path curvature on path acceleration)

• MD20603 $MC_CURV_EFFECT_ON_PATH_JERK[n>] (influence of path curvature on path jerk)

where index $n> =$

0 for DYNNORM
1 for DYNPOS
2 for DYNROUGH
3 for DYNSEMFIN
4 for DYNFINISH

**Note**

Writing the machine data **without an index** places the same value in all field elements of the relevant machine data.

Reading the machine data **without an index** always supplies the value of the field with index 0.

**Suppressing G commands**

It is recommended that G commands from G function group 59 (dynamic response mode for path interpolation) which are not intended for use should be suppressed via the following machine data:

MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[n>] (list of reconfigured NC commands)

If a suppressed G command is used, an alarm is displayed. This prevents machine data that has not been parameterized taking effect.

**Example**

The G commands **DYNPOS** and **DYNSEMFIN** can be suppressed with the following settings:

• MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[0]="DYNPOS"

• MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[1]=" "

• MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[2]="DYNSEMFIN"

• MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[3]=" "
4.4 Dynamic adaptations

References

You can find detailed information about programming the G commands from G function group 59 (dynamic response mode for path interpolation) in:

References:
Programming Manual, Job Planning; Section: Path traversing behavior

4.4.6 Free-form surface mode: Basic functions

Introduction

In applications in tool and mold making, it is important that the surfaces on the workpiece are as uniform as possible. This requirement is generally more important than the precision of the surface of the workpiece.

Workpiece surfaces which lack uniformity can be attributable to the following causes, for example:

- The part program for manufacturing the workpiece contains a non-uniform geometry. This, most notably, affects the profile of the curvature and torsion.

Note

The curvature \( k \) of a contour is the inverse of radius \( r \) of the adapted circle in a contour point \( (k = 1/r) \). The torsion is the change in curvature (first derivative).

As a result of the lack of uniformity in geometry, the machine's dynamic response limits are reached during processing of the part program, and needless deceleration and acceleration processes occur. Depending on the extent of the effective over-travel of the axes, this leads to different deviations in contours.

- Needless deceleration and acceleration processes can trigger machine vibrations which result in unwanted marks on the workpiece.

There are various options available for eliminating these causes:

- Use a CAD/CAM system
  The part programs generated by CAD/CAM systems contain a very uniform curvature and torsion profile, preventing needless reductions in path velocity.

- Specify the maximum path velocity in such a way that unwanted geometric fluctuations in the curvature and torsion profile have no effect.

Function

"Free-form surface mode: Basic functions" can be used to make the definition of path velocity limits insensitive to small geometric fluctuations in curvature and torsion without exceeding the machine's dynamic limits in terms of the acceleration and jerk of the axes.

This has the following advantages:

- Greater uniformity in the profile of the path velocity
- Greater uniformity in the surface of the workpiece
- Reduction in the processing time (if the dynamic response of the machine permits it)
Applications

The function is used to process workpieces which primarily comprise free-form surfaces.

Requirements

The function can only be activated if the requisite memory capacity is reserved during memory configuration:

\[
\text{MD28610 } \text{$MC\_MM\_PREPDYN\_BLOCKS = 10$}
\]

The value entered prescribes the number of blocks which have to be taken into consideration in the determination of the path velocity (velocity preparation).

A sensible value is "10".

If MD28610 has a value of "0", only the motions of the axes in a particular block are taken into consideration when determining the maximum velocity of the path for that block. If the geometry of neighboring blocks is also taken into consideration when determining the velocity of the path (value > 0), a more uniform profile in path velocity is achieved.

Activation/deactivation

The function can be switched on or off independently for every dynamic response mode (see Section "Dynamic response mode for path interpolation (Page 222)"):

\[
\text{MD20606 } \text{$MC\_PREPDYN\_SMOOTHING\_ON[<n>] = <value>$}
\]

<table>
<thead>
<tr>
<th>Index &lt;n&gt;</th>
<th>Dynamic response mode</th>
<th>&lt;value&gt;</th>
<th>Free-form surface mode: Basic functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Standard dynamic response settings (DYNNORM)</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>On</td>
</tr>
<tr>
<td>1</td>
<td>Positioning mode, tapping (DYNPOS)</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>On</td>
</tr>
<tr>
<td>2</td>
<td>Roughing (DYNROUGH)</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>On</td>
</tr>
<tr>
<td>3</td>
<td>Finishing (DYNSEMIFIN)</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>On</td>
</tr>
<tr>
<td>4</td>
<td>Smooth finishing (DYNFINISH)</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>On</td>
</tr>
</tbody>
</table>

Note

Due to the additional storage requirements, the function should only be activated in the relevant processing channels.

Parameterization

Change in the contour sampling factor
The secant error which occurs during the interpolation of curved contours is dependent on the following factors:

- Curvature
- Interpolator clock cycle (display in the MD10071 $MN_IPO_CYCLE_TIME)
- Velocity with which the relevant contour is traversed

The maximum possible secant error is defined for each axis in the machine data:

MD33100 $MA_COMPRESS_POS_TOL (maximum tolerance with compression)

If the set interpolator clock cycle is not sufficiently small, the max. path velocity may be reduced in the case of contours with greater curvature. This is necessary for ensuring that the surface of the workpiece is also produced with an adequate degree of precision in this case.

By changing the contour sampling factor, the time interval with which a curved contour is sampled in the interpolator (contour sampling time) can be set differently than the interpolator clock cycle. A contour sampling time which is shorter than the interpolator clock cycle can prevent a reduction in path velocity in the case of contours with greater curvature.

The contour sampling factor is set with the machine data:

MD10682 $MN_CONTOUR_SAMPLING_FACTOR

The effective contour sampling time is calculated as follows:

\[ T_s = f \times T_1 \]

where:
- \( T_s \) = Effective contour sampling time
- \( T_1 \) = Interpolator clock cycle
- \( f \) = Contour sampling factor (value from MD10682)

The default contour sampling factor is "1", i.e. the contour sampling time equals the interpolator clock cycle.

The contour sampling factor can be both greater or less than "1".

If a value less than "1" is set, monitoring of contour sampling precision is disabled.

The set sampling time must not be below the configured minimum contour sampling time:

MD10680 $MN_MIN_CONTOUR_SAMPLING_TIME

**Note**

MD10680 is specifically set for every controller model and cannot be changed.

### Programming

Depending on the setting in machine data MD20606 $MC_PREPDYN_SMOOTHING_ON, "Free-form surface mode: Basic functions" can be switched on and off in the part program by changing the active dynamic response mode.

**Example:**

By assigning the parameters MD20606 $MC_PREPDYN_SMOOTHING_ON[2-4] = 1 and MD20606 $MC_PREPDYN_SMOOTHING_ON[0-1] = 0, the function can be switched on via
the commands DYNROUGH, DYNSEMIFIN, and DYNFINISH and switched off via the commands DYNNORM and DYNNPOS.

See also

Rounding of tangential block transitions (G645) (Page 198)
Velocity-dependent jerk adaptation (axis-specific) (Page 268)
Free-form surface mode: Extension function (Page 206)

4.5 Compressor functions

4.5.1 Compression of blocks for linear and circular movement

Function

CAD/CAM systems generate a large number of linear and circular blocks, some of which with very short path lengths, to describe complex contours. The maximum possible path velocity is frequently limited by the block change time. As of a certain path velocity, not enough new traversing blocks can be prepared in the preprocessing and switched to the main run.

The compressor functions replace consecutive linear and circular blocks with polynomial blocks having the longest possible path lengths while achieving a parameterized contour precision. This has the following advantages:

- Reduction of the number of traversing blocks
- Increasing the path velocity
- Increasing the surface quality
- Continuous block transitions

COMPON, COMPCURV

The compressor functions COMPON and COMPCURV generate one polynomial block from up to ten consecutive linear blocks of the form: "G01 X... Y... Z... F...". The polynomial blocks of the compressor functions have the following properties:

- COMPON: Continuous velocity block transitions.
- COMPCURV: Continuous velocity and acceleration block transitions

COMPCAD

The compressor function COMPCAD can generate one polynomial block from theoretically any number of consecutive linear and circular blocks. The polynomial blocks have constant velocity and acceleration at the block transitions. Corners that are desirable are identified as such and taken into account.
COMPCAD is very CPU time and memory-intensive. It is therefore recommended that COMPCAD only be used where surface improvements were not successful by measures taken within the CAD/CAM program.

Common features

- If commands that are not traverse commands, e.g. auxiliary function output, are programmed in and between the traversing blocks to be compressed, compression will be interrupted.
- The maximum tolerable deviation of the calculated path to the programmed positions can be specified in machine data for all compressor functions. Unlike COMPON and COMPCURV, the parameterized tolerances are not used in different directions in neighboring paths with COMPCAD.

Parameterization

The compressor functions are parameterized in the following machine data and setting data:

**Axis-specific machine data**

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier $MA_-$</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD33100</td>
<td>COMPRESS_POS_TOL</td>
<td>Maximum permissible axis-specific path deviation with compression</td>
</tr>
</tbody>
</table>

**Channel-specific machine data**

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier $MC_-$</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD20170</td>
<td>COMPRESS_BLOCK_PATH_LIMIT</td>
<td>Maximum traversing length of a block for compression</td>
</tr>
<tr>
<td>MD20172</td>
<td>COMPRESS_VELO_TOL</td>
<td>Maximum permissible deviation from path feed for compression</td>
</tr>
<tr>
<td>MD20482</td>
<td>COMPRESSOR_MODE</td>
<td>Setting the mode of operation of the compressor</td>
</tr>
<tr>
<td>MD20484</td>
<td>COMPRESSOR_PERFORMANCE</td>
<td>Compressor performance</td>
</tr>
<tr>
<td>MD20485</td>
<td>COMPRESS_SMOOTH_FACTOR</td>
<td>Smoothing factor for compression</td>
</tr>
<tr>
<td>MD20486</td>
<td>COMPRESS_SPLINE_DEGREE</td>
<td>Compressor spline degree</td>
</tr>
<tr>
<td>MD20487</td>
<td>COMPRESS_SMOOTH_FACTOR_2</td>
<td>Smoothing factor 2 for compression</td>
</tr>
</tbody>
</table>

**Channel-specific setting data**

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier $SC_-$</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD42470</td>
<td>CRIT_SPLINE_ANGLE</td>
<td>Corner limit angle for COMPCAD</td>
</tr>
<tr>
<td>SD42475</td>
<td>COMPRESS_CONTUR_TOL</td>
<td>Maximum permissible contour deviation with compression</td>
</tr>
<tr>
<td>SD42476</td>
<td>COMPRESS_ORI_TOL</td>
<td>Maximum deviation of the tool orientation for compression</td>
</tr>
<tr>
<td>SD42477</td>
<td>COMPRESS_ORIROT_TOL</td>
<td>Maximum deviation of the tool rotation for compression</td>
</tr>
</tbody>
</table>
Note
Corner limit angle and compressor function COMPCAD

The corner limit angle for COMPCAD set via the setting data SD42470 $SC_CRIT_SPLINE_ANGLE is only used as an approximate measure for corner detection. By evaluating the plausibility, the compressor can also identify flatter block transitions as corners and larger angles as outliers.

Recommended settings

The following machine data has an effect on the compressor functions. We recommend setting the following values:

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier</th>
<th>Recommended value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD18360</td>
<td>$MN_MM_EXT_PROG_BUFFER_SIZE</td>
<td>100</td>
</tr>
<tr>
<td>MD20490</td>
<td>$MC_IGNORE_OVL_FACTOR_FOR_ADIS</td>
<td>1</td>
</tr>
<tr>
<td>MD28520</td>
<td>$MC_MM_MAX_AXISPOLY_PER_BLOCK</td>
<td>3</td>
</tr>
<tr>
<td>MD28530</td>
<td>$MC_MM_PATH_VELO_SEGMENTS</td>
<td>5</td>
</tr>
<tr>
<td>MD28540</td>
<td>$MC_MM_ARCLENGTH_SEGMENTS</td>
<td>10</td>
</tr>
<tr>
<td>MD28060</td>
<td>$MC_MM_IPO_BUFFER_SIZE</td>
<td>100</td>
</tr>
<tr>
<td>MD28070</td>
<td>$MC_MM_NUM_BLOCKS_IN_PREP</td>
<td>60</td>
</tr>
<tr>
<td>MD32310</td>
<td>$MA_MAX_ACCEL_OVL_FACTOR</td>
<td>&lt;Value for G64 operation&gt;</td>
</tr>
</tbody>
</table>

Programming

Syntax

COMPON
COMPCURV
COMPCAD
COMPOF

Meaning

<table>
<thead>
<tr>
<th></th>
<th>Activating the compressor function COMPON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effective:</td>
<td>Modal</td>
</tr>
<tr>
<td>COMPON:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Activating the compressor function COMPCURV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effective:</td>
<td>Modal</td>
</tr>
<tr>
<td>COMPCURV:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Activating the compressor function COMPCAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effective:</td>
<td>Modal</td>
</tr>
<tr>
<td>COMPCAD:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Deactivating the currently active compressor function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPOF:</td>
<td></td>
</tr>
</tbody>
</table>

See also

Tolerance and compression of G0 blocks (Page 235)
4.5.2 Compression of short spline blocks

Function

When spline blocks are generated to describe complex contours by means of CAD/CAM systems, spline blocks with very short path lengths occur between spline blocks with long path lengths. These force the control to significantly reduce the path velocity. The functions for the compression of short spline blocks generate new spline blocks with the longest possible path lengths.

Availability

<table>
<thead>
<tr>
<th>System</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINUMERIK 840D sl</td>
<td>Standard (basic scope)</td>
</tr>
<tr>
<td>SINUMERIK 828D</td>
<td>Option</td>
</tr>
</tbody>
</table>

Parameterization

The compression of short spline blocks can be activated for the following spline types:

- BSPLINE
- BSPLINE/ORICURVE
- CSPLINE

It is activated in the channel-specific machine data:

MD20488 $MC_SPLINE_MODE, bit \texttt{<n>=<value>}

<table>
<thead>
<tr>
<th>Bit</th>
<th>&lt;value&gt;</th>
<th>Meaning: Combine short spline blocks is</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Not active for BSPLINE</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Active for BSPLINE</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Not active for BSPLINE/ORICURVE</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Active for BSPLINE/ORICURVE</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>Not active for CSPLINE</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Active for CSPLINE</td>
</tr>
</tbody>
</table>

Supplementary conditions

- If commands that are not traverse commands, e.g. auxiliary function outputs, are programmed in and between the traversing blocks to be compressed, the spline blocks cannot be combined.
- The maximum number of blocks that can be combined into a program section in succession, depends on the parameterized size of the block memory available in the block preparation. MD28070 $MC_MM_NUM_BLOCKS_IN_PREP (number of blocks for block preparation)
Example

To achieve a higher path velocity when executing the traversing blocks, compression for short spline blocks is activated for BSPLINE interpolation:

MD20488 $MC_SPLINE_MODE, Bit 0 = 1

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G1 G64 X0 Y0 Z0 F1000</td>
<td>; Initial setting</td>
</tr>
<tr>
<td>N20 G91 F10000 BSPLINE</td>
<td>; Activation: B spline</td>
</tr>
<tr>
<td>N30 X0.001 Y0.001 Z0.001</td>
<td>; From here: Combine short spline blocks</td>
</tr>
<tr>
<td>N40 X0.001 Y0.001 Z0.001</td>
<td>...</td>
</tr>
</tbody>
</table>

4.5.3 Supplementary conditions

Orientation transformation (TRAORI)

When orientation transformation is active, the compressor functions COMPON, COMPCURV, and COMPCAD can also compress motion blocks for tool orientation and tool rotation. A detailed description can be found in:

Reference:
Function Manual, Special Functions; Chapter "F2: Multi-axis transformation" > "Orientation" > "Compression of the orientation"

Block search with calculation

- If the target block for block search type 2 or type 4 (block search with calculation to ...) is in a program section in which a compressor function (COMPON, COMPCURV, COMPCAD) is active, positions are approached on the path calculated by the compressor when the contour is approached again. These positions must precisely match the positions on the path programmed in the part program.

- If programmed blocks are eliminated from the part program during compression, these blocks will not be found at the target block in the block search ⇒ Alarm 15370 "Search target not found."
4.6 Contour/Orientation tolerance

Parameter assignment

**Machine data**
- Contour tolerance / orientation tolerance
  MD33100 $MA\_COMPRESS\_POS\_TOL[<axis>] = <value> (maximum tolerance with compression)
  
  Via the axis-specific machine data, the maximum permitted contour deviation (contour tolerance) or angle deviation of the tool orientation (orientation tolerance) of each axis is set. The machine data is effective with the following functions:
  - **Rounding:** G642, G643, G644, G645
  - **Compressor:** COMPON, COMPCURV, COMPCAD

  The higher the value, the more short blocks can be compressed into a long block.

  The machine data is not effective with the rounding function G641. The programmable distance to the block transition with ADIS= or ADISPOS= is used for G641.

- **Rounding mode**
  MD20480 $MC\_SMOOTING\_MODE (rounding behavior with G64x)

- **Compressor mode**
  MD20482 $MC\_COMPRESSOR\_MODE (mode of compression)

- **Rounding** G645
  MD33120 $MA\_PATH\_TRANS\_POS\_TOL (maximum contour deviation for rounding G645)
  Effective on rounding tangential, but not continuously curved block transitions (e.g. circle - straight line)

**Setting data**
- **Channel-specific contour tolerance**
  SD42465 $SC\_SMOOTHE\_CONTR\_TOL (maximum contour deviation)

- **Channel-specific orientation tolerance**
  SD42466 $SC\_SMOOTHE\_ORI\_TOL (maximum angular deviation of the tool orientation)

- **Channel-specific orientation tolerance with rounding with OST**
  SD42676 $SC\_ORI\_SMOOTHE\_TOL (tolerance for smoothing with orientation on rounding)

- **Channel-specific orientation tolerance with rounding with ORISON**
  SD42678 $SC\_ORISON\_TOL (tolerance for smoothing the orientation)

**Programming**

**Syntax**

CTOL = <value>
OTOL = <value>
ATOL[<axis>] = <value>
### Meaning

<table>
<thead>
<tr>
<th>CTOL: Programmable contour tolerance</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessing stop: No</td>
<td></td>
</tr>
<tr>
<td>Effective: Modal</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OTOL: Programmable orientation tolerance</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessing stop: No</td>
<td></td>
</tr>
<tr>
<td>Effective: Modal</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ATOL: Programmable axis-specific contour tolerance</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessing stop: No</td>
<td></td>
</tr>
<tr>
<td>Effective: Modal</td>
<td></td>
</tr>
</tbody>
</table>

<axis>: Name of the channel axis to which the programmed tolerance will apply

<value>: Tolerance value

<table>
<thead>
<tr>
<th>Value range:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>≥ 0: Tolerance value</td>
<td></td>
</tr>
<tr>
<td>&lt; 0: Resetting the programmed tolerance value ⇒ The tolerance value parameterized in the machine or setting data applies again.</td>
<td></td>
</tr>
</tbody>
</table>

### Notes

- The channel-specific tolerance values programmed with **CTOL** and **OTOL** have higher priority than the axis-specific tolerance values programmed with **ATOL**.
- With end of program or reset, the tolerance value parameterized in the machine or setting data is effective again.
System variables

Reading with preprocessing stop
Via the following system variables, the currently active tolerances can be read in the NC part program and synchronized action:

- $AC_CTOL
  Channel-specific contour tolerance effective when the current main run record was preprocessed.
  If no contour tolerance is effective, $AC_CTOL will return the root of the sum of the squares of the tolerances of the geometry axes.

- $AC_OTOL
  Channel-specific orientation tolerance effective when the current main run record was preprocessed.
  If no orientation tolerance is effective, $AC_OTOL will return the root of the sum of the squares of the tolerances of the orientation axes during active orientation transformation. Otherwise, it will return the value "-1."

- $AA_ATOL[<axis>]
  Axis-specific contour tolerance effective when the current main run record was preprocessed.
  If no contour tolerance is active, $AA_ATOL[<geometry axis>] returns the contour tolerance divided by the root of the number of geometry axes.
  If an orientation tolerance and an orientation transformation are active
  $AA_ATOL[<orientation axis>] will return the orientation tolerance divided by the root of the number of orientation axes.

Note
If now tolerance values have been programmed, the $A variables are not differentiated enough to distinguish the tolerance of the individual functions.

Circumstances like this can occur if the machine data and the setting data set different tolerances for compressor functions, smoothing and orientation smoothing. The system variables then return the greatest value occurring with the functions that are currently active. For example, if a compressor function is active with an orientation tolerance of 0.1° and ORISON orientation smoothing with 1°, the $AC_OTOL variable will return the value "1." If orientation smoothing is deactivated, $AC_OTOL will return the value "0.1."

Reading without preprocessing stop
Via the following system variables, the currently active tolerances can be read in the NC part program:

- $P_CTOL
  Currently active channel-specific contour tolerance.

- $P_OTOL
  Currently active channel-specific orientation tolerance.

- $PA_ATOL
  Currently active axis-specific contour tolerance.
Supplementary conditions

Programming
The tolerances programmed with CTOL, OTOL, and ATOL also affect functions that indirectly depend on these tolerances:

- Limiting the chord error in the setpoint value calculation
- The basic functions of the free-form surface mode

The following rounding functions are not affected by the programming of CTOL, OTOL, and ATOL:

- Rounding orientation with OSD
  OSD does not use a tolerance, it uses a distance from the block transition.

- Rounding with G644
  G644 is not used for processing, it is used for optimizing tool changes and other movements without machining.

- Rounding with G645
  G645 virtually always behaves like G642 and, thus, uses the programmed tolerances. The tolerance value from machine data MD33120 $MA\_PATH\_TRANS\_POS\_TOL is only used in uniformly tangential block transitions with a jump in curvature, e.g. a tangential circle/straight line transition. The rounding path at these points may also be located outside the programmed contour, where many applications are less tolerant. Furthermore, it generally takes a small, fixed tolerance to compensate for the sort of changes in curvature which need not concern the NC programmer.

4.7 Tolerance and compression of G0 blocks

Function

The function "Tolerance and compression of rapid traverse blocks (G00)" allows rapid traverse motion to be traversed faster.

It consists of the components:

1. Parameterization / programming of a tolerance factor for rapid traverse movements
   Using this factor, the tolerances for G0 motion can be set differently to the workpiece machining tolerances.

2. Compressing G0 blocks
   If the compression of G0 blocks is active while NC block compression is active, not only traversing blocks with programmed feedrate (G01) but also traversing blocks with G0 (rapid traverse) are compressed.
**G0 tolerance factor**

The tolerance factor for rapid traverse movements is only effective if both conditions are met:

1. One of the following functions is active:
   - Compressor function: COMPON, COMPCURV, or COMPCAD
   - Smoothing functions: G642 or G645
   - Orientation smoothing: OST
   - Orientation smoothing: ORISON
   - Smoothing for path-relevant orientation: ORIPATH

2. Several (≥ 2) consecutive G0 blocks in the part program.
   For a single rapid traverse block, the G0 tolerance factor is not effective, as the lower tolerance always applies (workpiece machining tolerance) at the transition from a non rapid traverse motion to a rapid traverse motion (and vice versa).

**Compression**

Compression of rapid traverse blocks is activated with the commands COMPON, COMPCURV, or COMPCAD. See Chapter: "Compression of blocks for linear and circular movement" (Page 227)

---

**Parameter assignment**

**G0 tolerance factor**

The G0 tolerance factor is set via the channel-specific machine data:

MD20560 $MC_G0_TOLERANCE_FACTOR (tolerance factor for G0)

The G0 tolerance factor can be both greater or less than 1.0. If the factor equals 1.0 (default value), the same tolerances are effective for non rapid traverse movements as for rapid traverse movement. Normally, the G0 tolerance factor is set to ≥ 1.0.

**Compressing G0 blocks**

The compression of G0 blocks is set for specific channels using the hundreds position in the machine data:

MD20482 $MC_COMPRESSOR_MODE = <value>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xx</td>
<td>Circular blocks and G0 blocks are not compressed.</td>
</tr>
<tr>
<td>1xx</td>
<td>Circular blocks are compressed. Only COMPCAD.</td>
</tr>
<tr>
<td>2xx</td>
<td>G0 blocks are compressed. See also MD20560 $MC_G0_TOLERANCE_FACTOR or NC command STOLF</td>
</tr>
<tr>
<td>3xx</td>
<td>Circular blocks and G0 blocks are compressed.</td>
</tr>
</tbody>
</table>

A detailed description of MD20482 can be found in:

**References**

Function Manual, Special Functions; Multi-Axis Transformations (F2); Section: "Compression of the orientation"
Programming

The tolerance factor set using MD20560 \$MC_G0_TOLERANCE_FACTOR can be temporarily overwritten by programming \texttt{STOLF} in the part program:

**Syntax**

\texttt{STOLF = <tolerance factor>}

**Meaning**

<table>
<thead>
<tr>
<th>STOLF:</th>
<th>Command to program the \texttt{G0} tolerance factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;tolerance factor&gt;:</td>
<td>\texttt{G0} tolerance factor</td>
</tr>
<tr>
<td></td>
<td>Value range: Tolerance factor &gt; 0.0</td>
</tr>
</tbody>
</table>

**Example**

In the following program section, the following tolerance factors apply alternately:

1. MD33100 \$MA_COMPRESS_POS_TOL (contour tolerance at corners) ⇒ \texttt{COMPCAD} (Page 227)
2. MD33120 PATH_TRANS_POS_TOL (contour tolerance for tangential transitions) ⇒ G645
3. MD20560 \$MC_G0_TOLERANCE_FACTOR
   Contour deviation = (MD20560 \$MC_G0_TOLERANCE_FACTOR) * (MD33100 \$MA_COMPRESS_POS_TOL)
4. Programmable contour tolerance for traversing blocks ⇒ \texttt{CTOL}
5. Programmable contour tolerance for rapid traverse blocks ⇒ \texttt{STOLF}

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPCAD G645 F10000</td>
<td>; Activation: Compressor function COMPCAD and G645</td>
</tr>
<tr>
<td>G1 X... Y... Z...</td>
<td>; Active tolerance factor □ 1. and 2.</td>
</tr>
<tr>
<td>X... Y... Z...</td>
<td>; Active tolerance factor □ 1. and 2.</td>
</tr>
<tr>
<td>X... Y... Z...</td>
<td>; Active tolerance factor □ 1. and 2.</td>
</tr>
<tr>
<td>G0 X... Y... Z...</td>
<td>; Active tolerance factor □ 2. and 3.</td>
</tr>
<tr>
<td>X... Y... Z...</td>
<td>; Active tolerance factor □ 2. and 3.</td>
</tr>
<tr>
<td>CTOL = 0.02</td>
<td>; Programmable contour tolerance for traversing blocks</td>
</tr>
<tr>
<td>STOLF = 4</td>
<td>; Programmable \texttt{G0} tolerance factor</td>
</tr>
<tr>
<td>G1 X... Y... Z...</td>
<td>; Active tolerance factor □ 4.</td>
</tr>
<tr>
<td>X... Y... Z...</td>
<td>; Active tolerance factor □ 4.</td>
</tr>
<tr>
<td>X... Y... Z...</td>
<td>; Active tolerance factor □ 4.</td>
</tr>
<tr>
<td>G0 X... Y... Z...</td>
<td>; Active tolerance factor □ 4. and 5. of 0.08 = 0.02 * 4</td>
</tr>
<tr>
<td>X... Y... Z...</td>
<td>; Active tolerance factor □ 4. and 5. of 0.08 = 0.02 * 4</td>
</tr>
</tbody>
</table>

The value in MD20560 is not changed by programming the tolerance factor. After a reset or end of part program, the value set using MD20560 is effective again.
System variables

The active G0 tolerance factor can be read via the following system variables.

- $AC_STOLF (active G0 tolerance factor)
- $P_STOLF (programmed G0 tolerance factor)

Reset response

After power-on reset, channel reset, or end of part program, the value parameterized in machine data MD20560 $MC_G0_TOLERANCE_FACTOR becomes effective again.

4.8 RESET behavior

MD20150

With the reset (channel or mode group reset), the initial setting parameterized channel-specifically becomes effective for all G groups:

MD20150 $MC_GCODE_RESET_VALUES (initial setting of the G groups)

The following G function groups are of relevance to "continuous-path mode, exact stop, LookAhead":

- Group 10: Exact stop - continuous-path mode
- Group 12: Block-change criterion for exact stop
- Group 21: Acceleration profile
- Group 30: NC block compression
- Group 59: Dynamic response mode for path interpolation

For detailed information on setting initial states, see Section "K1: Mode group, channel, program operation, reset response (Page 455)."

4.9 Supplementary conditions

4.9.1 Block change and positioning axes

If path axes are traversed in continuous path mode in a part program, traversing positioning axes can also simultaneously affect both the response of the path axes and the block change.

A detailed description of the positioning axes can be found in:

References:
Function Manual, Extended Functions; Positioning axes (P2)
4.9.2 Block change delay

Even if all path axes and special axes traversing in the part program block have satisfied their specific block transition criteria, the block change can still be delayed due to other unsatisfied conditions and/or active functions:

**Examples:**
- Missing auxiliary function acknowledgement by the PLC
- Non-existent following blocks
- Active function "Empty buffer"

**Effects**

If a block change cannot be executed in continuous path mode, all axes programmed in this part program block (except cross-block traversing special axes) are stopped. In this case, contour errors do not occur.

The stopping of path axes during machining can cause undercuts on the workpiece surface.

4.10 Data lists

4.10.1 Machine data

4.10.1.1 General machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10110</td>
<td>PLC_CYCLE_TIME_AVERAGE</td>
<td>Average PLC acknowledgment time</td>
</tr>
<tr>
<td>10680</td>
<td>MIN_CONTOUR_SAMPLING_TIME</td>
<td>Minimum contour sampling time</td>
</tr>
<tr>
<td>10682</td>
<td>CONTOUR_SAMPLING_FACTOR</td>
<td>Contour sampling factor</td>
</tr>
<tr>
<td>10712</td>
<td>NC_USER_CODE_CONF_NAME_TAB</td>
<td>List of reconfigured NC commands</td>
</tr>
<tr>
<td>12030</td>
<td>OVR_FACTOR_FEEDRATE</td>
<td>Evaluation of the path feed override switch</td>
</tr>
<tr>
<td>12100</td>
<td>OVR_FACTOR_LIMIT_BIN</td>
<td>Limit for binarycoded override switch</td>
</tr>
<tr>
<td>18360</td>
<td>MM_EXT_PROG_BUFFER_SIZE</td>
<td>FIFO buffer size for execution from external source (DRAM)</td>
</tr>
</tbody>
</table>

4.10.1.2 Channelspecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20150</td>
<td>GCODE_RESET_VALUES</td>
<td>Initial setting of the groups</td>
</tr>
<tr>
<td>20170</td>
<td>COMPRESS_BLOCK_PATH_LIMIT</td>
<td>Maximum traversing length of NC block for compression</td>
</tr>
<tr>
<td>20172</td>
<td>COMPRESS VELO_TOL</td>
<td>Maximum permissible deviation from path feed for compression</td>
</tr>
</tbody>
</table>
### 4.10 Data lists

#### 4.10.1.3 Axis/spindlespecific machine data

| Number | Identifier: $SM_A_\_\_\_\_\_|$ | Description |
|--------|-------------------------------|-------------|
| 32310  | MAX ACCEL OVL FACTOR          | Overload factor for axial velocity jumps |
| 32431  | MAX AX JERK                    | Maximum axial jerk when traversing along the path |
| 32432  | PATH TRANS JERK LIM            | Maximum axial jerk at the block transition in continuous-path mode |
| 32433  | SOFT ACCEL FACTOR              | Scaling of acceleration limitation for SOFT |
| 32434  | G00 ACCEL FACTOR               | Scaling of acceleration limitation for G00 |
| 32435  | G00 JERK FACTOR                | Scaling of axial jerk limitation for G00 |
| 32440  | LOOKAH FREQUENCY               | Smoothing limit frequency for LookAhead |
| 33100  | COMPRESS POS TOL               | Maximum deviation with compensation |
| 33120  | PATH TRANS POS TOL             | Maximum deviation when rounding with G645 |
4.10.2 Setting data

4.10.2.1 Channelspecific setting data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>35240</td>
<td>ACCEL_TYPE_DRIVE</td>
<td>DRIVE acceleration characteristic for axes on/off</td>
</tr>
<tr>
<td>36000</td>
<td>STOP_LIMIT_COARSE</td>
<td>Exact stop coarse</td>
</tr>
<tr>
<td>36010</td>
<td>STOP_LIMIT_FINE</td>
<td>Exact stop fine</td>
</tr>
<tr>
<td>36012</td>
<td>STOP_LIMIT_FACTOR</td>
<td>Exact stop coarse/fine factor and zero speed monitoring</td>
</tr>
<tr>
<td>36020</td>
<td>POSITIONING_TIME</td>
<td>Delay time exact stop fine</td>
</tr>
</tbody>
</table>

4.10.3 Signals

4.10.3.1 Signals from channel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>All axes stationary</td>
<td>DB21, ...DBX36.3</td>
<td>DB330x.DBX4.3</td>
</tr>
</tbody>
</table>

4.10.3.2 Signals from axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position reached with exact stop coarse</td>
<td>DB31, ...DBX60.6</td>
<td>DB390x.DBX0.6</td>
</tr>
<tr>
<td>Position reached with exact stop fine</td>
<td>DB31, ...DBX60.7</td>
<td>DB390x.DBX0.7</td>
</tr>
</tbody>
</table>
4.10 Data lists
B2: Acceleration

5.1 Brief description

5.1.1 General

Scope of functions

The Description of Functions covers the following sub-functions:

- Acceleration
- Jerk
- Kneeshaped acceleration characteristic

Acceleration and jerk

The effective acceleration and jerk can be optimally matched to the machine and machining situation concerned using axis- and channel-specific programmable maximum values, programmable acceleration profiles in part programs and synchronized actions, and dynamic adaptations and limitations.

Kneeshaped acceleration characteristic

The knee-shaped acceleration characteristic means that, in the case of machine axes featuring a motor (in particular stepper motors) with a torque characteristic that is highly dependent upon speed, acceleration can be set at the level required to ensure optimum utilization of the motor whilst at the same time protecting it against overload.

5.1.2 Features

Acceleration

Axis-specific functions:

- Programmable maximum acceleration value
- Acceleration profile that can be selected via part-program instruction: Acceleration without jerk limitation (BRISKA)
- Setting of maximum value using part-program instruction (ACC)
- Specific maximum value for programmed rapid traverse (G00).
- Specific maximum value for traverse with active jerk limitation
- Excessive acceleration for non-tangential block transitions
Channel-specific functions:
- Acceleration profile that can be selected via part-program instruction:
  Acceleration without jerk limitation (BRISK)
- Programmable constant travel time for the purpose of avoiding extreme sudden acceleration
- Programmable acceleration margin for overlaid traversing
- Adjustable acceleration limitation
- Adjustable acceleration for specific real-time events
- Programmable acceleration margin for radial acceleration

Jerk

Axis-specific functions:
- Acceleration profile that can be selected via part-program instruction:
  Acceleration with jerk limitation (SOFTA)
- Programmable maximum jerk value for single-axis interpolation
- Programmable maximum jerk value for path interpolation

Channel-specific functions:
- Acceleration profile that can be selected via part-program instruction:
  Acceleration with jerk limitation (SOFT)
- Adjustable jerk limitation
- Adjustable path jerk for specific real-time events
- Specific maximum value for programmed rapid traverse (G00)
- Excessive jerk for block transitions without constant curvature

Kneeshaped acceleration characteristic

A knee-shaped acceleration characteristic is parameterized using the following characteristic data:
- Maximum velocity \( v_{\text{max}} \)
- Maximum acceleration \( a_{\text{max}} \)
- Creep velocity \( v_{\text{red}} \)
- Creep acceleration \( a_{\text{red}} \)
- Nature of the acceleration reduction (constant, hyperbolic, linear)
5.2 Functions

5.2.1 Acceleration without jerk limitation (BRISK/BRISKA) (channel/axis-specific)

5.2.1.1 General Information

General Information

In the case of acceleration without jerk limitation (jerk = infinite) the maximum value is applied for acceleration immediately. As regard to acceleration with jerk limitation, it differs in the following respects:

- **Advantages**
  Shorter processing times with the same maximum values for velocity and acceleration.

- **Disadvantages**
  Increased load on the machine’s mechanical components and risk of inducing high-frequency and difficult-to-control mechanical vibrations.

**Acceleration profile**

![Acceleration profile diagram]

- $a_{\text{max}}$: Maximum acceleration value
- $v_{\text{max}}$: Maximum velocity value
- $t$: Time

Figure 5-1 Velocity and acceleration schematic for stepped acceleration profile
The following features of the acceleration profile can be identified from the figure above:

- **Time**: $t_0$
  Sudden acceleration from 0 to $+a_{\text{max}}$

- **Interval**: $t_0 - t_1$
  Constant acceleration with $+a_{\text{max}}$; linear increase in velocity

- **Time**: $t_1$
  Sudden acceleration from $2 \cdot a_{\text{max}}$ with immediate switchover from acceleration to braking

**Note**

The sudden acceleration can normally be avoided by specifying a constant velocity time (see Section "Constant travel time (channel-specific) (Page 248)").

- **Interval**: $t_1 - t_2$
  Constant acceleration with $-a_{\text{max}}$; linear decrease in velocity

### 5.2.1.2 Parameterization

#### Maximum axial acceleration for path motions

The maximum axial acceleration for path motions can be set for the specific technology for each machine axis via the following machine data:

$$\text{MD32300 $MA_{\text{MAX AX ACCEL}}[\text{parameter set index}]$}$$

With $\text{parameter set index} = 0, 1, 2 ...$ (max. parameter set number - 1)

For the technology-specific parameter sets, see Section "Dynamic response mode for path interpolation (Page 222)".

The path parameters are calculated by the path planning of the preprocessing so that the parameterized maximum values of the machine axes involved in the path are not exceeded.

**Note**

It is possible for the maximum value to be exceeded in connection with specific machining situations (see Section "Acceleration matching (ACC) (axis-specific) (Page 250)" and "Path acceleration for real-time events (channel-specific) (Page 253)").

#### Maximum axial acceleration for positioning axis motions

With positioning axis motions, one of the two following maximum values is effective depending on the set positioning axis dynamic response mode:

- $$\text{MD32300 $MA_{\text{MAX AX ACCEL 0}}$ (maximum axial acceleration for path motions in the dynamic response mode DYNNORM)}$$
- $$\text{MD32300 $MA_{\text{MAX AX ACCEL 1}}$ (maximum axial acceleration for path motions in the dynamic response mode DYNPPOS)}$$

The positioning axis dynamic response mode is set in the NC-specific machine data:
MD18960 $MN_POS_DYN_MODE = <mode>

<table>
<thead>
<tr>
<th>&lt;mode&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Effective maximum axial acceleration: MD32300 $MA_MAX_AX_ACCEL[ 0 ]</td>
</tr>
<tr>
<td>1</td>
<td>Effective maximum axial acceleration: MD32300 $MA_MAX_AX_ACCEL[ 1 ]</td>
</tr>
</tbody>
</table>

**Maximum axial acceleration for JOG motions**

For JOG mode, a JOG-specific maximum acceleration value can be configured for each machine axis (see Section "Acceleration and jerk for JOG motions (Page 282)").

**5.2.1.3 Programming**

**Path acceleration without jerk limitation (BRISK)**

**Syntax**

BRISK

**Functionality**

The BRISK part-program instruction is used to select the "without jerk limitation" acceleration profile for the purpose of path acceleration.

- **G group:** 21
- **Effective:** Modal

**Reset response**

The channel-specific initial setting is activated via a reset:

MD20150 $MC_GCODE_RESET_VALUES[20]

**Supplementary conditions**

If the acceleration profile is changed in a part program during machining (BRISK/SOFT) an exact stop is performed at the end of the block.

**Single-axis acceleration without jerk limitation (BRISKA)**

**Syntax**

BRISKA (axis{,axis})
**Function**

The BRISKA part-program command is used to select the "without jerk limitation" acceleration profile for single-axis movements (JOG, JOG/INC, positioning axis, reciprocating axis, etc.).

G group: -

Effectiveness: Modal

Axis:

- Value range: Axis name of the channel axes

**Axis-specific initial setting**

Acceleration without jerk limitation can be set as the axis-specific initial setting for single-axis movements:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE = FALSE

**Reset behavior**

The axis-specific initial setting is activated via a reset:

MD32420 $MA_JOG_AND_POS_ENABLE

---

5.2.2 Constant travel time (channel-specific)

5.2.2.1 General Information

**Overview**

In the case of acceleration without jerk limitation, sudden acceleration of $2 \times a_{\text{max}}$ occurs on switchover between acceleration and braking. In order to avoid this sudden acceleration, a channel-specific constant travel time can be programmed. The constant travel time defines the time taken to traverse between the acceleration and braking phases at constant velocity:

MD20500 $MC_{\text{CONST VELO MIN TIME}}$ (minimum time with constant velocity)

**Note**

The constant travel time is ineffective:

- Active function: Look Ahead
- In traversing blocks with a travel time that is less or equal to the interpolation cycle time.
1: Characteristic with constant travel time
2: Characteristic without constant travel time
$a_{\text{max}}$: Maximum acceleration value
$v_{\text{max}}$: Maximum velocity value
t: Time

Figure 5-2  Schematic for abrupt acceleration

The effect of the constant travel time can be seen from the figure above:

- Time: $t_1$
  End of acceleration phase with sudden acceleration $1 \cdot a_{\text{max}}$

- Interval: $t_1 - t_2$
  Acceleration 0; constant velocity using the parameterized constant travel time

- Time: $t_2$
  Start of braking phase with sudden acceleration $1 \cdot a_{\text{max}}$

The times $t_0$, $t_1'$ and $t_2'$ indicate the characteristic that would have been produced had no constant travel time been defined.

5.2.2.2 Parameterization

The constant travel time is parameterized for specific channels using machine data:

MD20500 $\text{SMC}_\text{CONST}_\text{VELO}_\text{MIN}_\text{TIME}$
(minimum time with constant velocity)
5.2.3 Acceleration matching (ACC) (axis-specific)

5.2.3.1 General Information

Function
Using the ACC command, the currently effective maximum axis acceleration parameterized in the acceleration-specific machine data can be reduced for a specific axis. The reduction is in the form of a percentage factor, which is specified when programming the command.

For instance, reducing the maximum possible axis acceleration in a machining segment can be used to prevent mechanical vibration as a result of high machining forces.

Effectiveness
Reducing the acceleration with ACC is effective for all interpolation types in the AUTOMATIC and MDI operating modes – and for the dry run feedrate.

Acceleration reduction is not effective:
- in the JOG mode
- During the machine function REF (reference point approach).
- If, as a result of a fault that has been detected, the axis is stopped with a fast stop (setpoint = 0).

5.2.3.2 Programming

Syntax
ACC[<axis>]=<reduction factor>
ACC[SPI(<spindle number>)]=<reduction factor>
ACC(S<spindle number>)=<reduction factor>

Meaning

<table>
<thead>
<tr>
<th>&lt;ACC&gt;</th>
<th>Command for the axis-specific reduction of the currently maximum possible acceleration, derived from the machine data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effectiveness: Modal</td>
<td></td>
</tr>
<tr>
<td>&lt;axis&gt;</td>
<td>Channel axis name of path axis</td>
</tr>
<tr>
<td>Data type: AXIS</td>
<td></td>
</tr>
<tr>
<td>Value range: Channel axis names</td>
<td></td>
</tr>
<tr>
<td>SPI(&lt;spindle number&gt;):</td>
<td>The SPI(...) function converts the spindle number into the corresponding channel axis name.</td>
</tr>
<tr>
<td>S&lt;spindle number&gt;:</td>
<td>Spindle name in the channel</td>
</tr>
</tbody>
</table>
Further information

System variable
The acceleration reduction (set using ACC), currently active in the channel, can be read on an axis-for-axis basis using:

$AA\_ACC[<axis>]$

Reset response
The acceleration reduction set using ACC can be kept after a channel reset or after the end of the program. The parameterization is performed via:

MD32320 $MA\_DYN\_LIMIT\_RESET\_MASK$

5.2.4 Acceleration margin (channel-specific)

5.2.4.1 General Information

General information
Under normal circumstances, preprocessing makes maximum use of the parameterized maximum values of the machine axes for the purpose of path acceleration. In order that an acceleration margin may be set aside for overlaid movements, e.g., within the context of the "Rapid lift away from the contour" function, path acceleration can be reduced by a programmable factor. When, for example, a factor of 0.2 is applied, preprocessing will only use 80% of the maximum possible path acceleration. 20% is set aside as an acceleration margin for overlaid movements.

5.2.4.2 Parameterization
Parameters for the acceleration margin are assigned for each channel by means of machine datum:

MD20610 $MC\_ADD\_MOVE\_ACCEL\_RESERVE$
(acceleration margin for overlaid motions)

5.2.5 Path-acceleration limitation (channel-specific)

5.2.5.1 General Information

General Information
To enable a flexible response to the machining situations concerned, setting data can be used to limit the path acceleration calculated during preprocessing for specific channels:

SD42500 $SC\_SD\_MAX\_PATH\_ACCEL$ (maximum path acceleration)
The value specified in the setting data is only taken into account if it is smaller than the path acceleration calculated during preprocessing.

The limitation must be activated for specific channels using setting data:

SD42502 $SC_IS_SD_MAX_PATH_ACCEL = TRUE

5.2.5.2 Parameterization

Parameterization is carried out for specific channels using setting data:

SD42500 $SC_SD_MAX_PATH_ACCEL (maximum path acceleration)

SD42502 $SC_IS_SD_MAX_PATH_ACCEL (activation of path-acceleration limitation)

5.2.5.3 Programming

Limit value

Syntax

$SC_SD_MAX_PATH_ACCEL = limit value

Functionality

The path-acceleration limitation can be adjusted for the situation by programming the setting data.

Limit value:

- Value range: \( \geq 0 \)
- Unit: m/s\(^2\)

Application:

- Part program
- Static synchronized action

Switch ON/OFF

Syntax

$SC_IS_SD_MAX_PATH_ACCEL = value

Functionality

The path-acceleration limitation can be activated/deactivated by programming the setting data.
Parameter: Value
- Value range: TRUE, FALSE

Application:
- Part program
- Static synchronized action

5.2.6 Path acceleration for real-time events (channel-specific)

5.2.6.1 General Information

General Information
So that no compromise has to be made between machining-optimized acceleration on the one hand and time-optimized acceleration in connection with the following real-time events on the other:
- NC Stop / NC Start
- Changing the feedrate override
- Changing the velocity default for "safely reduced velocity" within the context of the "Safety Integrated" function

For the real-time events mentioned above, the path acceleration can be specified using a channel-specific system variable:

\[
\text{\$AC\_PATHACC} = \text{path acceleration}
\]

Real-time event acceleration will only be active for the duration of the change in velocity in respect of one of the real-time events specified above.

Limitation

If the specified path acceleration exceeds the capabilities of the machine axes that are of relevance for the path, a limit will be imposed on the path acceleration within the controller so that the resulting axial acceleration \(a_{\text{res}}\) is restricted to less than 2x the parameterized maximum axial value \(a_{\text{max}}\).

\[
a_{\text{res}} = 2 \cdot a_{\text{max}}, \text{ with } a_{\text{max}} = \text{MD32300 $MA\_MAX\_AX\_ACCEL}$
\]

Note
Path acceleration for real-time events is enabled, irrespective of the radial acceleration.
Effectiveness

<table>
<thead>
<tr>
<th>Effective</th>
<th>Real-time event acceleration is only enabled in AUTOMATIC and MDA operating modes in conjunction with the following real-time events:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• NC Stop / NC Start</td>
</tr>
<tr>
<td></td>
<td>• Override changes</td>
</tr>
<tr>
<td></td>
<td>• Changing the velocity default for &quot;safely reduced velocity&quot; within the context of the &quot;Safety Integrated&quot; function</td>
</tr>
<tr>
<td>Not effective</td>
<td>Path acceleration for real-time events is ineffective for changes in path velocity that are attributable to path planning during preprocessing for the channel, such as contour curvatures, corners, kinematic transformation limitations, etc.</td>
</tr>
<tr>
<td></td>
<td>Real-time-event path acceleration is ineffective if the programmed value is smaller than the path acceleration calculated during preprocessing for the path section concerned.</td>
</tr>
</tbody>
</table>

Programming

For information about programming system variables in the part program or synchronized actions, see Section "Programming (Page 254)".

5.2.6.2 Programming

Syntax

$AC_PATHACC = path acceleration

Functionality

Real-time-event path acceleration is set via the channel-specific system variables.

Parameter: Path acceleration

- Value range: Path acceleration ≥ 0
- Unit: m/s²

Deactivation: $AC_PATHACC = 0

Application:

- Part program
- Static synchronized action

Reset response

Real-time-event path acceleration is deactivated on reset.

Supplementary conditions

Programming $AC_PATHACC in the part program automatically triggers a preprocessing stop with REORG (STOPRE).
5.2.7 Acceleration with programmed rapid traverse (G00) (axis-specific)

5.2.7.1 General Information

Frequently, the acceleration for the machine axes involved in the machining process must be set lower than the machine’s performance capability officially allows because of the supplementary conditions associated with the specific process concerned.

For time-optimized traversing of the machine axes with programmed rapid traverse (part-program instruction \texttt{G00}), a specific maximum value can be programmed for the axis-specific acceleration.

\textbf{JOG setup mode}

This function does not affect acceleration in respect of a rapid traverse override in JOG setup mode.

5.2.7.2 Parameterization

The maximum value for axis-specific acceleration with programmed rapid traverse is parameterized (\texttt{G00}) using the axis-specific machine data:

\begin{itemize}
  \item MD32434 $\text{MA\_G00\_ACCEL\_FACTOR}$ (scaling of the acceleration limitation with \texttt{G00})
\end{itemize}

This is used to generate the maximum value for axis-specific acceleration with programmed rapid traverse (\texttt{G00}) that is taken into account by the path planning component during preprocessing:

\begin{align*}
\text{Acceleration}[\text{axis}] &= \\
&= \text{MD32300 $\text{MA\_MAX\_AX\_ACCEL}$} \times \text{MD32434 $\text{MA\_G00\_ACCEL\_FACTOR}$}
\end{align*}
5.2.8 Acceleration with active jerk limitation (SOFT/SOFTA) (axis-specific)

5.2.8.1 General Information

Function

Compared with acceleration without jerk limitation, acceleration with jerk limitation results in a certain degree of time loss, even when the same maximum acceleration value is used. To compensate for this time loss, a specific maximum value can be programmed for the axis-specific acceleration as far as traversing of the machine axes with active jerk limitation (SOFT/SOFTA) is concerned.

The maximum value for acceleration with active jerk limitation is parameterized using a factor calculated in relation to the axis-specific maximum value. This is used to generate the maximum value for axis-specific acceleration with active jerk limitation that is taken into account by the path planning component during preprocessing:

\[
\text{Acceleration[axis]} = \text{MD32300 $\text{MA}_\text{MAX}_\text{AX}_\text{ACCEL}$ \times \text{MD32433 $\text{MA}_\text{SOFT}_\text{ACCEL}\_\text{FACTOR}$}
\]

5.2.8.2 Parameterization

The maximum value for acceleration with active jerk limitation (SOFT/SOFTA) is parameterized using the axis-specific machine data:

MD32434 $\text{MA}_\text{SOFT}_\text{ACCEL}\_\text{FACTOR}$

(scaling of the acceleration limitation with SOFT)

5.2.9 Excessive acceleration for non-tangential block transitions (axis-specific)

5.2.9.1 General Information

Function

In the case of non-tangential block transitions (corners), the programmable controller may have to decelerate the geometry axes significantly in order to ensure compliance with the parameterized axis dynamics. For the purpose of reducing/avoiding deceleration in connection with non-tangential block transitions, a higher level of axis-specific acceleration can be enabled.

Excessive acceleration is parameterized using a factor calculated in relation to the axis-specific maximum value. This is used to generate the maximum value for axis-specific acceleration with non-tangential block transitions that is taken into account by the path planning component during preprocessing:

\[
\text{Acceleration[axis]} = \text{MD32300 $\text{MA}_\text{MAX}_\text{AX}_\text{ACCEL}$ \times \text{MD32310 $\text{MA}_\text{MAX}_\text{ACCEL}_\text{OVL}\_\text{FACTOR}$}
\]
5.2.9.2 Parameterization

Excessive acceleration for non-tangential block transitions is parameterized using the axis-specific machine data:
MD32310 $MA_MAX_ACCEL_OVL_FACTOR
(overload factor for velocity jumps)

5.2.10 Acceleration margin for radial acceleration (channel-specific)

5.2.10.1 General Information

Overview

In addition to the path acceleration (tangential acceleration), radial acceleration also has an effect on curved contours. If this is not taken into account during parameterization of the path parameters, the effective axial acceleration during acceleration and deceleration on the curved contour can, for a short time, reach 2x the maximum value.

Effective axial acceleration =
Path acceleration + radial acceleration =
2 * ( MD32300 $MA_MAX_AX_ACCEL )

Figure 5-3 Radial and path acceleration on curved contours

The channel-specific machine data:
MD20602 $MC_CURV_EFFECT_ON_PATH_ACCEL
(influence of path curvature on dynamic path response)
can be used to set the proportion of the axis-specific acceleration that is to be taken into account for radial acceleration.

When, for example, a value of 0.75 is applied, 75% of the axis-specific acceleration will be made available for radial acceleration and 25% for path acceleration.
The corresponding maximum values are generally calculated as follows:

Radial acceleration =  
$$MD20602 \, MC\_CURV\_EFFECT\_ON\_PATH\_ACCEL \times MD32300 \, MA\_MAX\_AX\_ACCEL$$

Path acceleration =  
$$(1 - MD20602 \, MC\_CURV\_EFFECT\_ON\_PATH\_ACCEL) \times MD32300 \, MA\_MAX\_AX\_ACCEL$$

Example

The following machine parameters apply:

- MD32300 $MA\_MAX\_AX\_ACCEL$ for all geometry axes: 3 m/s
- Maximum path velocity with a path radius of 10 mm due to mechanical constraints of the machine: 5 m/min.

The radial acceleration is calculated as follows:

$$a_{radial} = \frac{v^2_{path} \, [m/min]}{r \, [mm] \times 3.6 \, [m/s^2]} = \frac{5^2}{10 \times 3.6} = 0.694 \, m/s^2$$

The acceleration margin is set as follows:

$$\frac{a_{radial} \, [m/s^2]}{MD32300 \, MA\_MAX\_AX\_ACCEL [m/s^2]} = \frac{0.694}{3} = 0.23$$

Linear motions

The acceleration margin referred to above is ineffective in the case of linear motions (linear interpolation) without active kinematic transformation.

5.2.10.2 Parameterization

The proportion of maximum available axis acceleration to be taken into account as an acceleration margin for radial acceleration on curved contours is parameterized using the channel-specific machine data:

$$MD20602 \, MC\_CURV\_EFFECT\_ON\_PATH\_ACCEL$$

(influence of path curvature on dynamic path response)
5.2.11 Jerk limitation with path interpolation (SOFT) (channel-specific)

5.2.11.1 General Information

Overview
As far as the functionality described in the rest of this document is concerned, constant acceleration, i.e., acceleration with jerk limitation (jerk = infinite value), is the assumed acceleration profile. In the case of acceleration with jerk limitation, linear interpolation is applied in respect of acceleration from 0 to the maximum value.

Advantages
Minimal load on the machine's mechanical components and low risk of high-frequency and difficult-to-control mechanical vibrations thanks to constant excessive acceleration.

Disadvantages
Longer machining times compared with stepped acceleration profile when the same maximum velocity and acceleration values are used.
The following features of the acceleration profile can be identified from the figure above:

- Interval: $t_0 - t_1$
  - Constant jerk with $+r_{\text{max}}$: linear increase in acceleration; quadratic increase in velocity

- Interval: $t_1 - t_2$
  - Constant acceleration with $+a_{\text{max}}$: linear increase in velocity

- Interval: $t_2 - t_3$
  - Constant jerk with $-r_{\text{max}}$: linear decrease in acceleration; quadratic decrease in excessive velocity until maximum value $+v_{\text{max}}$ is reached

- Interval: $t_3 - t_4$
  - Constant jerk with $+r_{\text{max}}$: linear increase in braking acceleration; quadratic decrease in velocity
5.2.11.2 Parameterization

Maximum jerk value for path motions (axis-specific)

The maximum axial jerk for path motions can be set for the specific technology for each machine axis via the following machine data:

MD32431 $MA_MAX_AX_JERK[<parameter set index>]

With <parameter set index> = 0, 1, 2 ... (max. parameter set number - 1)

For the technology-specific parameter sets, see Section “Dynamic response mode for path interpolation (Page 222)”. The path parameters are calculated by the path planning of the preprocessing so that the parameterized maximum values of the machine axes involved in the path are not exceeded.

Note

It is possible for the maximum value to be exceeded in connection with specific machining situations (see Section “Path jerk for real-time events (channel-specific) (Page 265)”).

Maximum jerk value for path motions (channel-specific)

In addition to the axis-specific setting, the maximum jerk value can also be specified as channel-specific path parameter via the following machine data:

MD20600 $MC_MAX_PATH_JERK (path-related maximum jerk)

In order to exclude the mutual influencing of axis and channel-specific maximum jerk values, the channel-specific maximum value must be set to a value greater than the axial maximum values.

5.2.11.3 Programming

Syntax

SOFT

Functionality

The SOFT part-program instruction is used to select the acceleration profile with jerk limitation for the traversing operations of geometry axes in the channel.
G group: 21
Effective: Modal

Reset response

The channel-specific initial setting is activated via a reset:

\[ \text{MD20150 \$MC\_GCODE\_RESET\_VALUES}[20] \]

Boundary conditions

If the acceleration mode is changed in a part program during machining (BRISK \leftrightarrow SOFT), a block change is performed at the point of transition with an exact stop at the end of the block, even in continuous-path mode.

5.2.12 Jerk limitation with single-axis interpolation (SOFTA) (axis-specific)

5.2.12.1 Parameterization

Initial setting for axial jerk limitation

Acceleration with jerk limitation can be set as the axial initial setting:

\[ \text{MD32420 \$MA\_JOG\_AND\_POS\_JERK\_ENABLE} == \text{TRUE} \]

Maximum axial jerk for positioning axis motions

When traversing positioning axes with active jerk limitation, the value from one of the following machine data takes effect as maximum axial jerk:

- MD32430 \$MA\_JOG\_AND\_POS\_MAX\_JERK (maximum axial jerk for positioning axis motions)
- MD32431 \$MA\_MAX\_AX\_JERK [0] (maximum axial jerk for path motions in the dynamic response mode DYNNORM)
- MD32431 \$MA\_MAX\_AX\_JERK [1] (maximum axial jerk for path motions in the dynamic response mode DYNPOS)

The machine data to be used is determined by the set positioning axis dynamic response mode:

\[ \text{MD18960 \$MN\_POS\_DYN\_MODE} = \text{<mode>} \]

<table>
<thead>
<tr>
<th>&lt;mode&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MD32430 $MA_JOG_AND_POS_MAX_JERK</td>
</tr>
<tr>
<td>1</td>
<td>MD32431 $MA_MAX_AX_JERK[1]</td>
</tr>
</tbody>
</table>
Maximum axial jerk for JOG motions

For JOG mode, a JOG-specific maximum jerk value can be configured for each machine axis (see Section "Acceleration and jerk for JOG motions (Page 282)").

5.2.12.2 Programming

Syntax

SOFTA (Axis {Axis})

Functionality

The SOFTA part-program command is used to select acceleration with jerk limitation for single-axis movements (positioning axis, reciprocating axis, etc.)

G group: -
Effectiveness: Modal
Axis:
• Value range: Axis name of the channel axes

Axis-specific initial setting

Acceleration with jerk limitation can be set as the axis-specific initial setting for single-axis movements:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE = TRUE

Reset behavior

The axis-specific initial setting is activated via a reset:

MD32420 $MA_JOG_AND_POS_ENABLE

5.2.13 Path-jerk limitation (channel-specific)

5.2.13.1 General Information

Overview

To enable a flexible response to the machining situations concerned, setting data can be used to limit the path jerk calculated during preprocessing for specific channels:

SD42510 $SC_SD_MAX_PATH_JERK (maximum path jerk)

The value specified in the setting data is only taken into account in the channel if it is smaller than the path jerk calculated during preprocessing.
The limitation must be activated for specific channels using setting data:
SD42512 $SC\_IS\_SD\_MAX\_PATH\_JERK = TRUE

5.2.13.2 Parameterization
Parameterization is carried out for specific channels using setting data:
SD42510 $SC\_SD\_MAX\_PATH\_JERK (maximum path jerk)
SD42512 $SC\_IS\_SD\_MAX\_PATH\_JERK (activation of path-jerk limitation)

5.2.13.3 Programming

Maximum path jerk

Syntax
$$SC\_SD\_MAX\_PATH\_JERK = jerk\ value$$

Functionality
The path-jerk limitation can be adjusted for the situation by programming the setting data.

Jerk value:
- Value range: ≥ 0
- Unit: m/s³

Application:
- Part program
- Static synchronized action

Switch ON/OFF

Syntax
$$SC\_IS\_SD\_MAX\_PATH\_JERK = value$$

Functionality
The path-jerk limitation can be activated/deactivated by programming the setting data.
Parameter: Value
- Value range: TRUE, FALSE

Application:
- Part program
- Static synchronized action

5.2.14 Path jerk for real-time events (channel-specific)

5.2.14.1 General Information

Overview
So that no compromise has to be made between machining-optimized jerk on the one hand and time-optimized jerk in connection with the following real-time events on the other:
- NC Stop / NC Start
- Changing the feedrate override
- Changing the velocity default for "safely reduced velocity" within the context of the "Safety Integrated" function

for the real-time events mentioned, the path jerk can be specified using a channel-specific system variable:
$AC_PATHJERK = path jerk

Path jerk for real-time events will only be active for the duration of the change in velocity in respect of one of the real-time events specified above.

Limitation
As the jerk is not a physical variable of any relevance to the drive, no limit is imposed on the jerk set.
Effectiveness

<table>
<thead>
<tr>
<th>Effective</th>
<th>Path jerk for real-time events is only enabled in AUTOMATIC and MDA operating modes in conjunction with the following real-time events:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• NC Stop / NC Start</td>
</tr>
<tr>
<td></td>
<td>• Override changes</td>
</tr>
<tr>
<td></td>
<td>• Changing the velocity default for &quot;safely reduced velocity&quot; within the context of the &quot;Safety Integrated&quot; function</td>
</tr>
</tbody>
</table>

| Not effective     | Path jerk for real-time events is ineffective for changes in the path velocity that are attributable to path planning during preprocessing for the channel, such as contour curvatures, corners, kinematic transformation limitations, etc. |
|                   | Path jerk for real-time events is ineffective if the programmed value is smaller than the path jerk calculated during preprocessing for the path section concerned. |

Programming

For the purpose of setting the jerk for real-time events in accordance with the acceleration, the system variables can be set as follows:

\[ \$AC\_PATHJERK = \frac{\$AC\_PATHACC}{\text{smoothing time}} \]

- \$AC\_PATHACC: Path acceleration \( [m/s^2] \)
- Smoothing time: Freely selectable, e.g. 0.02 s

For information about programming system variables in the part program or synchronized actions, see Section "Programming (Page 266)".

5.2.14.2 Programming

Syntax

\[ \$AC\_PATHJERK = \text{path jerk} \]

Functionality

The path jerk for real-time events is set via the channel-specific system variables.

Jerk value:
- Value range: Path jerk \( \geq 0 \)
- Unit: \( m/s^3 \)

Application:
- Part program
- Static synchronized action

Reset behavior

The function is deactivated on reset.
Boundary conditions

Programming $AC_PATHJERK in the part program automatically triggers a preprocessing stop with REORG ($STOPRE).

5.2.15 Jerk with programmed rapid traverse (G00) (axis-specific)

5.2.15.1 General Information

Overview

Frequently, the maximum jerk for the machine axes involved in the machining process must be set lower than the machine’s performance capability officially allows because of the supplementary conditions associated with the specific process concerned.

For time-optimized traversing of the machine axes with programmed rapid traverse (part-program instruction G00), a specific maximum value can be programmed for the axis-specific jerk.

JOG setup mode

This function does not affect jerk in respect of a rapid traverse override in JOG setup mode.

5.2.15.2 Parameterization

The maximum value for axis-specific jerk with programmed rapid traverse is parameterized (G00) using the axis-specific machine data:

MD32434 $MA_G00_ACCEL_FACTOR
(scaling of the acceleration limitation with G00)

This is used to generate the maximum value for axis-specific jerk with programmed rapid traverse (G00) that is taken into account by the path planning component during preprocessing:

Jerk[axis] = MD32431 $MA_MAX_AX_JERK * MD32435 $MA_G00_JERK_FACTOR
5.2.16 Excessive jerk for block transitions without constant curvature (axis-specific)

5.2.16.1 General Information

Overview

In the case of block transitions without constant curvature (e.g. straight line > circle), the programmable controller has to decelerate movement of the geometry axes significantly in order to ensure compliance with the parameterized axis dynamics. For the purpose of reducing/avoiding deceleration in connection with block transitions without constant curvature, a higher level of axis-specific jerk can be enabled.

The excessive jerk is parameterized using a dedicated axis-specific maximum value.

5.2.16.2 Parameterization

The excessive jerk for block transitions without constant curvature is parameterized using the axis-specific machine data:

MD32432 $MA_PATH_TRANS_JERK_LIM
(excessive jerk for block transitions without constant curvature)

5.2.17 Velocity-dependent jerk adaptation (axis-specific)

Function

The dynamic path response results from the parameterized, constant axial maximum values for velocity, acceleration and jerk of the axes involved in the path:

- MD32000 $MA_MAX_AX_VELO (maximum axis velocity)
- MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration)
- MD32431 $MA_MAX_AX_JERK (max. axial jerk for path motion)

For contours with non-constant curvature (torsion), as for example in connection with free-form surfaces, fluctuations in the path velocity, particularly in the upper velocity range, can result mainly due to the axial jerk. The fluctuations of the path velocity lead to adverse affects in the surface quality.

The influence of the axial jerk on the path velocity is decreased for contours with non-constant curvature through a velocity-dependent increase of the permissible axial jerk. Fluctuations in the path velocity can be avoided with the appropriate parameterization.

The velocity-dependent increase of the permissible axial jerk has no effect on the maximum possible path acceleration and path jerk. These result from the constant axial maximum values parameterized in the in the machine data even when jerk adaptation is active.

As both curvature and torsion are zero in the case of linear motion, the velocity-dependent jerk adaptation has no effect with linear motions.
Availability

The "velocity-dependent jerk adaptation" function is available independent of the function "Free-form surface mode: Basic functions (Page 224)".

Parameterization

The "Velocity-dependent jerk adaptation" function is parameterized with the following machine data:

- MD32437 $MA_AX_JERK_VEL0[<n>] = <threshold value lower>
  Lower velocity threshold of the jerk adaptation. Velocity-dependent jerk adaptation takes effective as of this velocity.
  The lower velocity threshold can be set separately via index n for each dynamic response mode (see Section "Dynamic response mode for path interpolation (Page 222)":)

- MD32438 $MA_AX_JERK_VEL1[<n>] = <threshold value upper>
  Upper velocity threshold of the jerk adaptation. The velocity-dependent jerk reaches its maximum value $j_{max}$ parameterized with MD32439 $MA_MAX_AX_JERK_FACTOR$ at this velocity.
  The upper velocity threshold can be set separately via index n for each dynamic response mode (see Section "Dynamic response mode for path interpolation (Page 222)":)

- MD32439 $MA_MAX_AX_JERK_FACTOR = <factor>
  Factor for the parameterization of the maximum velocity-dependent jerk $j_{max}$ on reaching the upper velocity threshold MD32438 $MA_AX_JERK_VEL1[<n>]$:
  \[ j_{max} = (MD32431$MA_MAX_AX_JERK$) \times (MD32439$MA_MAX_AX_JERK_FACTOR$) \]
  The velocity-dependent jerk adaptation is active at a value > 1.0.
  The velocity-dependent jerk adaptation is inactive at a value = 1.0.

Figure 5-5 Axial jerk as a function of the axis velocity
Note
The velocity-dependent jerk adaptation is only active, if:
MD32439 $MA\_MAX\_AX\_JERK\_FACTOR > 1.0

Example
Example of parameter assignment:
- MD32437 $MA\_AX\_JERK\_VEL0 = 3000 \text{ mm/min}$
- MD32438 $MA\_AX\_JERK\_VEL1 = 6000 \text{ mm/min}$
- MD32439 $MA\_MAX\_AX\_JERK\_FACTOR[AX1] = 2.0$
- MD32439 $MA\_MAX\_AX\_JERK\_FACTOR[AX2] = 3.0$
- MD32439 $MA\_MAX\_AX\_JERK\_FACTOR[AX3] = 1.0$

Effect
- The velocity-dependent jerk adaptation becomes active for the 1st and 2nd axis, while the function for the 3rd axis is not active.
- The parameterized jerk is effective at axis velocities in the range of 0 to 3000 mm/min.
- The maximum jerk is linearly increased for axis velocities in the range 3000 mm/min to 6000 mm/min.
- The maximum permitted jerk of the 1st axis is, for axis velocities greater than 6000 mm/min, increased by a factor of 2 - for the 2nd axis, by factor of 3.
- The parameterized values apply in each dynamic response mode.

5.2.18 Jerk filter (axis-specific)

5.2.18.1 General Information

Overview
In certain application scenarios, e.g. when milling free-form surfaces, it may be beneficial to smooth the position setpoint characteristics of the machine axes. This enables surface quality to be improved by reducing the mechanical vibrations generated in the machine.

For the purpose of smoothing the position setpoint characteristic of a machine axis, a jerk filter can be activated at position controller level, independently of the channel- and axis-specific jerk limitations taken into account at interpolator level.

The effect of the jerk filter must be as strong as possible without having an unacceptable impact on contour accuracy. The filter should also have as "balanced" a smoothing effect as possible, i.e. if the same contour is traversed forwards and backwards, the contour smoothed by the filter should be as similar as possible in both directions.
To enable the jerk filter to be optimally matched to the machine conditions, various filter modes are available:

- 2nd-order filter (PT2)
- Sliding mean value generation
- Bandstop filter

**Mode: 2nd-order filter**

Owing to the fact that it is a simple low-pass filter, "2nd-order filter" mode can only meet the requirements specified above where relatively small filter time constants (around 10 ms) are concerned. When used in conjunction with larger time constants, impermissible contour deviations are soon manifest. The effect of the filter is relatively limited.

This filter mode offers advantages if very large filter time constants are needed and contour accuracy is only of secondary importance (e.g. positioning axes).

For historical reasons, this filter mode is set as the default.

**Mode: Sliding mean value generation**

Where minimal contour deviations are required, filter time constants within the range of 20-40 ms can be set using the "sliding mean value generation" filter mode. The smoothing effect is largely symmetrical.

The display of the calculated servo gain factor \(K_v\) in the user interface, shows smaller values than would normally be appropriate for the filter. The contour accuracy is in fact higher than the displayed servo gain factor \(K_v\) appears to suggest.

When changing from "2nd-order filter" to "sliding mean value generation" filter mode, the displayed servo gain factor \(K_v\) may therefore drop (with identical filter time constant), even though there is an improvement in contour accuracy.

**Mode: Bandstop filter**

The bandstop filter is a 2nd-order filter in terms of numerator and denominator:

\[
H(s) = \frac{s^2}{(2 \pi f_2)^2} + \frac{2sD_Z}{(2 \pi f_2)} + \frac{s^2}{(2 \pi f_n)^2} + \frac{2sD_N}{2 \pi f_n}
\]

where:
- \(f_2\): Numerator natural freq.
- \(f_n\): Denominator natural freq.
- \(D_Z\): Numerator damping
- \(D_N\): Denominator damping
Since a vibration-capable filter setting is not expected to yield useful results in any case, as with the jerk filter’s “2nd-order filter" (PT2) low-pass filter (PT2) mode there is no setting option for the denominator damping $D_N$. The denominator damping $D_N$ is permanently set to 1.

The bandstop filter can be parameterized in two different ways:

- Real bandstop filter
- Bandstop filter with additional amplitude response increase/decrease at high frequencies

**Real bandstop filter**

The real bandstop filter is applied when identical numerator and denominator natural frequencies are selected:

$$f_z = f_N = f_{\text{block}}$$ (blocking frequency)

If numerator damping setting = 0 is selected, the blocking frequency is equivalent to complete attenuation. In this case the 3 dB bandwidth is calculated as follows:

$$f_{3\,\text{dB\ bandwidth}} = 2 \cdot f_{\text{block}}$$

If instead of complete attenuation, a reduction by a factor of $k$ is all that is required, then numerator damping should be selected in accordance with $k$. In this case the above formula for calculating the 3 dB bandwidth no longer applies.

**Bandstop filter with additional amplitude response increase/decrease at high frequencies**

In this case, the numerator and denominator natural frequencies are set to different values. The numerator natural frequency determines the blocking frequency.

By selecting a lower/higher denominator natural frequency than the numerator natural frequency, you can increase/decrease the amplitude response at high frequencies. An amplitude response increase at high frequencies can be justified in most cases, as the controlled system generally possesses a lowpass characteristic itself, i.e. the amplitude response drops at high frequencies anyway.

**Supplementary conditions**

If too high a numerator natural frequency is selected, the filter is disabled. In this case the limiting frequency $f_{z_{\text{max}}}$ depends on the position-control cycle:

$$f_{z_{\text{max}}} = \frac{1}{2 \cdot \pi \cdot T_{\text{z_{min}}}} = \frac{1}{2 \cdot \pi \cdot T_{\text{position-control cycle}}}$$ (Shannon Theorem)

### 5.2.18.2 Parameterization

**Activation**

The jerk filter is activated using the machine data:

`MD32400 $MA_AX_JERK_ENABLE (axial jerk limitation)`

The jerk filter is active in all operating modes and with all types of interpolation.
Filter mode

The filter mode is selected via the machine data:

MD32402 $MA_AX_JERK_MODE

<table>
<thead>
<tr>
<th>Value</th>
<th>Filter mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2nd-order filter</td>
</tr>
<tr>
<td>2</td>
<td>Sliding mean value generation</td>
</tr>
<tr>
<td>3</td>
<td>Bandstop filter</td>
</tr>
</tbody>
</table>

Time constant

The time constant for the axial jerk filter is set with the machine data:

MD32410 $MA_AX_JERK_TIME

The jerk filter is only effective when the time constant is greater than a position-control cycle.
5.2.19  Kneeshaped acceleration characteristic curve

5.2.19.1  Adaptation to the motor characteristic curve

Function
Various types of motor, particularly stepper motors, have a torque characteristic that is highly dependent upon speed and shows a steep decrease in torque in the upper speed range. To ensure optimum utilization of the motor characteristic curve, it is necessary to reduce the acceleration once a certain speed is reached.

Simulation of torque characteristic
For the purpose of simulating the torque characteristic of the motor characteristic curve, the machine data:
MD35242 $MA_ACCEL_REDUCTION_TYPE = characteristic
can be used to select various types of characteristic curve:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Constant characteristic</td>
</tr>
<tr>
<td>1</td>
<td>Hyperbolic characteristic</td>
</tr>
<tr>
<td>2</td>
<td>Linear characteristic</td>
</tr>
</tbody>
</table>

Figure 5-6  Torque characteristic curve of a motor with torque characteristic that is highly dependent upon speed
The following figures show typical velocity and acceleration characteristic curves for the respective types of characteristic:

**Constant characteristic**

![Constant Characteristic Diagram](image)

Figure 5-7  Acceleration and velocity characteristic with acceleration reduction: 0 = constant

**Hyperbolic characteristic**

![Hyperbolic Characteristic Diagram](image)

Figure 5-8  Acceleration and velocity characteristic with acceleration reduction: 1 = hyperbolic

**Linear characteristic**

![Linear Characteristic Diagram](image)

Figure 5-9  Acceleration and velocity characteristic with acceleration reduction: 2 = linear
The key data for the characteristic curves equate to:

\[
\begin{align*}
    v_{\text{max}} &= \$\text{MA\_MAX\_AX\_VELO} \\
    v_{\text{red}} &= \$\text{MA\_ACCEL\_REDUCTION\_SPEED\_POINT} \times \$\text{MA\_MAX\_AX\_VELO} \\
    a_{\text{max}} &= \$\text{MA\_MAX\_AX\_ACCEL} \\
    a_{\text{red}} &= (1 - \$\text{MA\_ACCEL\_REDUCTION\_FACTOR}) \times \$\text{MA\_MAX\_AX\_ACCEL}
\end{align*}
\]

5.2.19.2 Effects on path acceleration

Function

The path acceleration characteristic curve is generated on the basis of the types of characteristic for the axes that are of relevance for the path. If axes with different types of characteristic curve are interpolated together, the acceleration profile for the path acceleration will be determined on the basis of the reduction type that is most restrictive.

The following order of priorities applies, whereby 1 = top priority:

1. Acceleration reduction: 0 = constant characteristic
2. Acceleration reduction: 1 = hyperbolic characteristic
3. Acceleration reduction: 2 = linear characteristic
4. No acceleration reduction effective

A situation, whereby no acceleration reduction is active, arises for example when:

\[
\begin{align*}
    \text{MD35220} \ $\text{MA\_ACCEL\_REDUCTION\_SPEED\_POINT} &= 1 \\
    \text{MD35230} \ $\text{MA\_ACCEL\_REDUCTION\_FACTOR} &= 0
\end{align*}
\]

Note

Machine axes featuring stepper motor and DC drive can be interpolated together.

5.2.19.3 Substitute characteristic curve

Function

If the programmed path cannot be traversed with the acceleration curve parameterized in the machine data of the participating axes, e.g. if kinematic transformation and acceleration types \textsc{brisk} or \textsc{soft} are active, a substitute curve is generated by reducing the dynamic limit values. The reduced dynamic limit values are calculated to ensure that the substitute characteristic curve provides the best possible compromise between maximum velocity and constant acceleration.
Substitute characteristic curve with linear path sections

Limitation to this value is applied if the programmed path velocity is greater than that at which 15% of the maximum acceleration capacity is still available \(v_{15\%a}\). Consequently, 15% of the maximum acceleration capacity/motor torque always remains available, whatever the machining situation.

![Substitute path characteristic curve: Linear path](image)

- **1** Normal range
- **2** Reducing range
- **3** Locked area

\(a_{\text{ers}}\): Substitute characteristic curve constant acceleration
\(a_{15\%}\): Minimal constant acceleration
\(a_{\text{ers}} = 0.15 \times (a_{\text{max}} - a_{\text{red}}) + a_{\text{red}}\)

\(v_{\text{ers}}\): Substitute characteristic curve velocity
\(v_{\text{prog}}\): Programmed velocity
\(v_{15\%a}\): Velocity at \(a_{15\%}\)

Figure 5-10 Substitute path characteristic curve: Linear path
**Substitute characteristic curve with curved path sections**

In the case of curved path sections, normal and tangential acceleration are considered together. The path velocity is reduced so that only up to 25% of the speed-dependent acceleration capacity of the axes is required for normal acceleration. The remaining 75% of the acceleration capacity is set aside for the tangential acceleration, i.e., deceleration/acceleration on the path.

![Diagram showing characteristic curve for curved path sections](image)

Figure 5-11 Substitute path characteristic curve: Curved path

1. Normal range
2. Reducing range

- $a_n$: Normal acceleration
- $a_{ers}$: Substitute characteristic curve constant acceleration
- $v_{ers}$: Substitute characteristic curve velocity
- $r$: Path radius

**Block transitions with continuous-path mode**

If continuous-path mode is active, non-tangential block transitions result in axial velocity jumps when the programmed path velocity is used for traversing.

As a result, the path velocity is controlled in such a way that prevents any axial velocity proportion from exceeding the creep velocity $v_{red}$ at the time of the block transition.

**Deceleration ramp with continuous-path mode and LookAhead**

In the case of consecutive traversing blocks with short paths, an acceleration or deceleration operation may be spread over several part program blocks.

In such a situation the "LookAhead" function also takes into account the parameterized speed-dependent acceleration characteristic.
5.2.19.4 Parameterization

The knee-shaped acceleration characteristics can be activated specific to the machine axis via the machine data:

MD35240 $MA_ACCEL_TYPE_DRIVE = TRUE

The knee-shaped acceleration characteristic curve is parameterized for specific axes using machine data:

MD32000 $MA_MAX_AX VELO (maximum axis velocity)
MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT (speed for reduced acceleration)
MD35230 $MA_ACCEL_REDUCTION_FACTOR (reduced acceleration)
MD32300 $MA_MAX_AX ACCEL (Maximum axis acceleration)
MD35242 $MA_ACCEL_REDUCTION_TYPE (type of acceleration reduction: 0 = constant, 1 = hyperbolic, 2 = linear)
5.2.19.5 Programming

Channel-specific activation (DRIVE)

Syntax

DRIVE

Functionality

The knee-shaped characteristic curve is activated for path acceleration using the DRIVE part-program instruction.

G group: 21
Effective: Modal

Reset response

The channel-specific default setting is activated via a reset:

MD20150 $MC_GCODE_RESET_VALUES[20]

Dependencies

If the knee-shaped acceleration characteristic curve is parameterized for a machine axis, then this becomes the default acceleration profile for all traversing operations.

If the effective acceleration profile is changed for a specific path section using the SOFT or BRISK part-program instructions, then an appropriate substitute characteristic curve with lower dynamic limit values is used in place of the knee-shaped acceleration characteristic curve.

The knee-shaped acceleration characteristic curve can be reactivated by reprogramming DRIVE.

Axis-specific activation (DRIVEA)

Syntax

DRIVEA (Axis {Axis})

Functionality

The knee-shaped characteristic curve is activated for all single-axis interpolations (positioning axis, reciprocating axis, etc.) for specific axes using the part-program command.

G group: -
Effectiveness: Modal
Axis:
- Value range: Axis name of the channel axes

Reset behavior

The channel-specific default setting is activated via a reset:

MD20150 $MC_GCODE_RESET_VALUES[20]

Dependencies

If the knee-shaped acceleration characteristic curve is parameterized for a machine axis, then this becomes the default acceleration profile for all traversing operations.

If the effective acceleration profile is changed for a specific axis using the `SOFTA` or `BRISKA` part-program commands, then an appropriate substitute characteristic curve is used in place of the knee-shaped acceleration characteristic curve.

It is possible to switch back to the knee-shaped acceleration characteristic curve for a specific axis by programming `DRIVEA`.

5.2.19.6 Boundary conditions

Single axis interpolation

After activating the knee-shaped acceleration characteristics in case of single-axis interpolations (positioning axis, oscillating axis, manual motion, etc.), the machine axis is traversed exclusively in the mode `DRIVEA`.

It is not possible to switch over the acceleration profile via the following part program instructions:
- Abrupt acceleration changes (`BRISKA`)
- Acceleration with jerk limitation (`SOFTA`)

Path interpolation

If for a machine axis involved in a programmed path the knee-shaped acceleration characteristic parameterized without the part program instruction `DRIVE` is active, then a substitute characteristic curve with reduced dynamic limiting values is determined for the path.

Kinematic transformation

The knee-shaped acceleration characteristic is not considered in an active kinematic transformation. With internal control, a switchover is done to acceleration without jerk limitation (`BRISK`) and a substitute characteristic curve becomes active for the path acceleration.
5.2.20 Acceleration and jerk for JOG motions

In order to avoid unwanted machine motions in JOG mode, separate axial acceleration and jerk limit values can be specified for JOG motion.

It is also possible to limit acceleration and jerk channel-specifically for manual traversing of geometry and orientation axes. This enables better handling of the kinematics that generate Cartesian motions entirely via rotary axes (robots).

5.2.20.1 Parameterization

Axial limitation of acceleration and jerk

Maximum axial acceleration for JOG motions
The maximum axial acceleration for JOG motion can be specified for each machine axis via the machine data:

MD32301 $MA_JOG_MAX_ACCEL
With MD32301 = 0, the value from MD32300 $MA_MAX_AX_ACCEL is effective instead of the JOG-specific maximum value.

Maximum axial jerk for JOG motions
The maximum axial jerk for JOG motion can be specified for each machine axis via the machine data:

MD32436 $MA_JOG_MAX_JERK
With MD32436 = 0, the value from MD32430 $MA_JOG_AND_POS_MAX_JERK is effective instead of the JOG-specific maximum value.

Note
MD32436 $MA_JOG_MAX_JERK is only effective when the axial jerk limitation in JOG mode has been enabled for the machine axes to be traversed:
MD32420 $MA_JOG_AND_POS_JERK_ENABLE [<axis>] == TRUE
This is also possible by programming SOFTA(<Axis1>,<Axis2>, ... ) in the part program.

Channel-specific limitation of acceleration and jerk

Maximum acceleration when manually traversing geometry axes
The maximum acceleration when manually traversing geometry axes can be specified for each channel via the machine data:

MD21166 $MC_JOG_ACCEL_GEO [<geometry axis>]
With <geometry axis> = 0, 1, 2
With MD21166 = 0, the axis-specific limit value from MD32301 $MA_JOG_MAX_ACCEL is effective instead of the channel-specific acceleration limitation.

**Note**

With MD21166 $MC_JOG_ACCEL_GEO [<geometry axis>], there is no direct limitation to MD32300 $MA_MAX_AX_ACCEL.

**Note**

When a transformation is active, MD32300 $MA_MAX_AX_ACCEL determines the maximum possible axial acceleration.

**Maximum jerk when manually traversing geometry axes**

The maximum jerk when manually traversing geometry axes in the SOFT acceleration mode (acceleration with jerk limitation) can be specified for each channel via the machine data:

MD21168 $MC_JOG_JERK_GEO [<geometry axis>]

With <geometry axis> = 0, 1, 2

With MD21168 = 0, the axis-specific limit value from MD32436 $MA_JOG_MAX_JERK is effective instead of the channel-specific jerk limitation.

**Note**

MD21168 $MC_JOG_JERK_GEO is only effective when the axial jerk limitation in JOG mode has been enabled for the basic machine axes:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE [<axis>] == TRUE

**Maximum jerk when manually traversing orientation axes**

The maximum jerk when manually traversing orientation axes can be specified for each channel via the machine data:

MD21158 $MC_JOG_JERK_ORI [<orientation axis>]

For MD21158 to take effect, the channel-specific jerk limitation for the manual traversing of orientation axes must be enabled via the following machine data:

MD21159 $MC_JOG_JERK_ORI_ENABLE == TRUE

**Note**

Orientation axes are not affected by the machine data MD32301 $MA_JOG_MAX_ACCEL and MD32436 $MA_JOG_MAX_JERK.
5.2.20.2 Supplementary conditions

Path override / overlaid motion

With path override / overlaid motion (e.g. DRF), the JOG-specific maximum values for acceleration and jerk (MD32301 $MA_JOG_MAX_ACCEL and MD32436 $MA_JOG_MAX_JERK) are not effective. Instead, the values for positioning axis motions are effective:

- **Acceleration:**
  - MD32300 $MA_MAX_AX_ACCEL [0] (maximum axial acceleration for path motions in the dynamic response mode DYNNORM)
  See also Section "Acceleration without jerk limitation (BRISK/BRISKA) (channel/axis-specific) (Page 245)".

- **Jerk**
  - MD32430 $MA_JOG_AND_POS_MAX_JERK (maximum axial jerk for positioning axis motions)
    or (with G75/751):
  - MD32431 $MA_MAX_AX_JERK [0] (maximum axial jerk for path motions in the dynamic response mode DYNNORM)
  See also Section "Jerk limitation with single-axis interpolation (SOFTA) (axis-specific) (Page 262)".

**Note**

Only the dynamic response mode DYNNORM is always effective for JOG mode.

Behavior for manual traversing of geometry axes with active rotation

When manually traversing geometry axes in the SOFT acceleration mode (acceleration with jerk limitation), the value from MD32436 $MA_JOG_MAX_JERK or MD32430 $MA_JOG_AND_POS_MAX_JERK is also used with active rotation or active orientable toolholder.

Part program instruction SOFTA/BRISKA/DRIVEA

The part program instruction SOFTA(<Axis1>,<Axis2>, ... ) is also effective in JOG mode, i.e. the maximum axial jerk from MD32436 $MA_JOG_MAX_JERK is effective for the specified axes when traversing in JOG mode (exactly as when setting MD32420 $MA_JOG_AND_POS_JERK_ENABLE [<axis>] == TRUE).

**Note**

On the other hand the SOFT part program instruction has no effect on the JOG mode.
As with SOFTA, the part program instructions BRISKA and DRIVEA are also effective in JOG mode, i.e. the acceleration is without jerk limitation, even when MD32420 $MA_JOG_AND_POS_JERK_ENABLE is set to "TRUE" for the relevant machine axes.

**Note**
The manual traversing of orientation axes is not affected by BRISKA/SOFTA/DRIVEA.

### 5.3 Examples

#### 5.3.1 Acceleration

##### 5.3.1.1 Path velocity characteristic

The following example shows the path velocity characteristic, based on programmed traversing motion and the actions initiated in a part program segment.

**Part program extract**

```
; Synchronized action: Acceleration switchover depending on fast input 1 ($A_IN[1]):
N53 ID=1 WHEN $A_IN[1] == 1 DO $AC_PATHACC = 2.*$MA_MAX_AX_ACCEL[X]
; Synchronized action: Test override profile (simulation of external interventions):
N54 ID=2 WHENEVER ($AC_TIMEC > 16) DO $AC_OVR=10
N55 ID=3 WHENEVER ($AC_TIMEC > 30) DO $AC_OVR=100

; Approach
N1000 G0 X0 Y0 BRISK
N1100 TRANS Y=-50
N1200 AROT Z=30 G642
; Contour
N2100 X0 Y0
N2200 X = 70 G1 F10000 RNDM=10 ACC[X]=30 ACC[Y]=30
N2300 Y = 70
N2400 X0
N2500 Y0
```
Path velocity characteristic

Figure 5-13 Switching between path acceleration specified during preprocessing and real-time acceleration

**Acceleration profile: BRISK**

1: Accelerate to 100% of path velocity (F10000) in accordance with acceleration default: ACC (N2200...)
2: Brake to 10% of path velocity as a result of override modification ($AC_OVR) in accordance with real-time acceleration $AC_PATHACC (N53/N54...)
3: Accelerate to 100% of path velocity as a result of override modification ($AC_OVR) in accordance with real-time acceleration $AC_PATHACC (N53/N55...)
4: Brake to block end velocity for intermediate smoothing block in accordance with acceleration default: ACC (N2200...)
5: Speed limitation as a result of smoothing (see 9)
6: Accelerate to 100% of path velocity ($AC_OVR) in accordance with acceleration default: ACC (N2300...)
7: Decelerate as a result of override modification at a rate of acceleration that is in accordance with real-time acceleration $AC_PATHACC (N53/N54...)
8: Accelerate to 100% of path velocity as a result of override modification ($AC_OVR) in accordance with real-time acceleration $AC_PATHACC (N53/N55...)
9: Intermediate block inserted within the control as a result of the programmed smoothing (RNDM) (N2200...)

5.3.2 Jerk

5.3.2.1 Path velocity characteristic

The following example shows the characteristic of path velocity and jerk based on programmed traversing motion and the actions initiated in a part program segment.

Part program extract

Program code

; Setting of path acceleration and path jerk in the event of external intervention:
N0100 $AC_PATHACC = 0.0
N0200 $AC_PATHJERK = 4.0 * ($SMA_MAX_AX_JERK[X] + $SMA_MAX_AX_JERK[Y]) / 2.0
**Program code**

; Synchronized action: varying the override (simulation of external interventions)

N0530 ID=1 WHENEVER ($AC_TIMEC > 16) DO $AC_OVR=10
N0540 ID=2 WHENEVER ($AC_TIMEC > 30) DO $AC_OVR=100

; Approach
N1000 G0 X0 Y0 SOFT
N1100 TRANS Y=-50
N1200 AROT Z=30 G642

; Contour
N2100 X0 Y0
N2200 X = 70 G1 F10000 RNDM=10
N2300 Y = 70
N2400 X0
N2500 Y0

**Characteristic of path velocity and jerk**

![Characteristic of path velocity and jerk](image)

**Acceleration profile: SOFT**

1: Jerk according to $MA_MAX_AX_JERK[..]
2: Jerk according to $AC_PATHJERK
3: Jerk according to $MA_MAX_AX_JERK[..] (approach block end velocity)
4: Velocity limiting using arcs
5: Jerk according to $AC_PATHJERK

Figure 5-14 Switching between path jerk specified during preprocessing and $AC_PATHJERK
5.3.3 Acceleration and jerk

The following example shows the characteristic of velocity and acceleration of the X axis based on the programmed traversing motion of the part program extract. Further, which of the velocity and acceleration-relevant machine data are decisive for which section of the contour.

Part program extract

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N90 F5000 SOFT G64</td>
<td>; Continuous-path mode, jerk-limited acceleration</td>
</tr>
<tr>
<td>N100 G0 X0 Y0 Z0</td>
<td>; Rapid traverse</td>
</tr>
<tr>
<td>N110 G1 X10</td>
<td>; Straight line</td>
</tr>
<tr>
<td>N120 G3 CR=5 X15 Y5</td>
<td>; Circular arc, radius 5 mm, block transition: Tangential</td>
</tr>
<tr>
<td>N130 G3 CR=10 X5 Y15</td>
<td>; Circular arc, radius 10 mm, block transition: Tangential</td>
</tr>
<tr>
<td>N140 G1 X-5 Y17.679</td>
<td>; Straight line, 15° kink</td>
</tr>
</tbody>
</table>

Contour

![Contour of the part program extract](image)

Figure 5-15 Contour of the part program extract
Velocity and acceleration characteristic

5.3.4 Knee-shaped acceleration characteristic curve

5.3.4.1 Activation

The example illustrates how the knee-shaped acceleration characteristic curve is activated on the basis of the machine data and a part program extract.

Machine data

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT[X]$</td>
<td>= 0.4</td>
</tr>
<tr>
<td>MD35230 $MA_ACCEL_REDUCTION_FACTOR[X]$</td>
<td>= 0.85</td>
</tr>
<tr>
<td>MD35242 $MA_ACCEL_REDUCTION_TYPE[X]$</td>
<td>= 2</td>
</tr>
<tr>
<td>MD35240 $MA_ACCEL_TYPE_DRIVE[X]$</td>
<td>= TRUE</td>
</tr>
<tr>
<td>MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT[Y]$</td>
<td>= 0.0</td>
</tr>
<tr>
<td>MD35230 $MA_ACCEL_REDUCTION_FACTOR[Y]$</td>
<td>= 0.6</td>
</tr>
<tr>
<td>MD35242 $MA_ACCEL_REDUCTION_TYPE[Y]$</td>
<td>= 1</td>
</tr>
<tr>
<td>MD35240 $MA_ACCEL_TYPE_DRIVE[Y]$</td>
<td>= TRUE</td>
</tr>
</tbody>
</table>
### Machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN_\text{_}</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>18960</td>
<td>POS_DYN_MODE</td>
<td>Type of positioning axis dynamic response</td>
</tr>
</tbody>
</table>

### Channel-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC_\text{_}</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20150</td>
<td>GCODE_RESET_VALUES</td>
<td>Initial setting of the G groups</td>
</tr>
<tr>
<td>20500</td>
<td>CONST_VELO_MIN_TIME</td>
<td>Minimum time with constant velocity</td>
</tr>
</tbody>
</table>
### Number | Identifier: \$MC\_ | Description
--- | --- | ---
20600 | MAX\_PATH\_JERK | Path-related maximum jerk
20602 | CURV\_EFFECT\_ON\_PATH\_ACCEL | Influence of path curvature on path dynamic response
20610 | ADD\_MOVE\_ACCEL\_RESERVE | Acceleration reserve for overlaid motions
21158 | JOG\_JERK\_ORI | Maximum jerk of the orientation axes when traversing in JOG
21159 | JOG\_JERK\_ORI\_ENABLE | Initial setting of the channel-specific jerk limitation of orientation axes for traversing in JOG mode
21166 | JOG\_ACCEL\_GEO | Maximum acceleration rate of the geometry axes when traversing in JOG
21168 | JOG\_JERK\_GEO | Maximum jerk of the geometry axes when traversing in JOG

#### 5.4.1.3 Axis/spindlespecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>32000</td>
<td>MAX_AX_VELO</td>
<td>Maximum axis velocity</td>
</tr>
<tr>
<td>32300</td>
<td>MAX_AX_ACCEL</td>
<td>Maximum axis acceleration</td>
</tr>
<tr>
<td>32301</td>
<td>JOG_MAX_ACCEL</td>
<td>Maximum acceleration in JOG mode</td>
</tr>
<tr>
<td>32310</td>
<td>MAX_ACCEL_OVL_FACTOR</td>
<td>Overload factor for velocity jump</td>
</tr>
<tr>
<td>32320</td>
<td>DYN_LIMIT_RESET_MASK</td>
<td>Reset behavior of dynamic limits</td>
</tr>
<tr>
<td>32400</td>
<td>AX_JERK_ENABLE</td>
<td>Axial jerk limitation</td>
</tr>
<tr>
<td>32402</td>
<td>AX_JERK_MODE</td>
<td>Filter type for axial jerk limitation</td>
</tr>
<tr>
<td>32410</td>
<td>AX_JERK_TIME</td>
<td>Time constant for axial jerk filter</td>
</tr>
<tr>
<td>32420</td>
<td>JOG_AND_POS_JERK_ENABLE</td>
<td>Initial setting for axial jerk limitation</td>
</tr>
<tr>
<td>32430</td>
<td>JOG_AND_POS_MAX_JERK</td>
<td>Axial jerk for single axis motion</td>
</tr>
<tr>
<td>32431</td>
<td>MAX_AX_JERK</td>
<td>Maximum axial jerk at the block transition in continuous-path mode</td>
</tr>
<tr>
<td>32432</td>
<td>PATH_TRANS_JERK_LIM</td>
<td>Max. axial jerk of a geometry axis at block boundary</td>
</tr>
<tr>
<td>32433</td>
<td>SOFT_ACCEL_FACTOR</td>
<td>Scaling of acceleration limitation for SOFT</td>
</tr>
<tr>
<td>32434</td>
<td>G00_ACCEL_FACTOR</td>
<td>Scaling of acceleration limitation for G00</td>
</tr>
<tr>
<td>32435</td>
<td>G00_JERK_FACTOR</td>
<td>Scaling of axial jerk limitation for G00</td>
</tr>
<tr>
<td>32436</td>
<td>JOG_MAX_JERK</td>
<td>Maximum axial jerk for JOG motion</td>
</tr>
<tr>
<td>32437</td>
<td>AX_JERK_VEL0</td>
<td>First velocity threshold for velocity-dependent jerk adaptation</td>
</tr>
<tr>
<td>32438</td>
<td>AX_JERK_VEL1</td>
<td>Second velocity threshold for the velocity-dependent jerk adaptation</td>
</tr>
<tr>
<td>32439</td>
<td>MAX_AX_JERK_FACTOR</td>
<td>Factor to set the maximum jerk for higher velocities (velocity-dependent jerk adaptation)</td>
</tr>
<tr>
<td>35220</td>
<td>ACCEL_REDUCTION_SPEED_POINT</td>
<td>Speed for reduced acceleration</td>
</tr>
<tr>
<td>35230</td>
<td>ACCEL_REDUCTION_FACTOR</td>
<td>Reduced acceleration</td>
</tr>
<tr>
<td>35240</td>
<td>ACCEL_TYPE_DRIVE</td>
<td>DRIVE acceleration characteristic for axes on/off</td>
</tr>
<tr>
<td>35242</td>
<td>ACCEL_REDUCTION_TYPE</td>
<td>Type of acceleration reduction</td>
</tr>
</tbody>
</table>
5.4.2 Setting data

5.4.2.1 Channelspecific setting data

| Number | Identifier: $SC_\_
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>42500</td>
<td>SD_MAX_PATH_ACCEL</td>
</tr>
<tr>
<td>42502</td>
<td>IS_SD_MAX_PATH_ACCEL</td>
</tr>
<tr>
<td>42510</td>
<td>SD_MAX_PATH_JERK</td>
</tr>
<tr>
<td>42512</td>
<td>IS_SD_MAX_PATH_JERK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_PATHACC</td>
<td>Path acceleration for real-time events</td>
</tr>
<tr>
<td>$AC_PATHJERK</td>
<td>Path jerk for real-time events</td>
</tr>
</tbody>
</table>
6.1 Brief description

Function

With the "Travel to fixed stop" function, moving machine parts, e.g. tailstock or sleeve, can be traversed so that they can apply a defined torque or force with respect to other machine parts over any time period.

Characteristics

- Programmability using commands in the part program or synchronized action:
  - Activating and deactivating the "Travel to fixed stop" function
  - Setting the clamping torque
  - Setting the monitoring window
- Manual intervention option using setting data:
  - Activating and deactivating the "Travel to fixed stop" function
  - Setting the clamping torque
  - Setting the monitoring window
- Presettings using machine data
  - Clamping torque
  - Monitoring window
- Identifying the function status via NC/PLC interface signals
- Enable or acknowledgment options via NC/PLC interface signals
- Reading the reference and actual status of the function via system variable
- "Travel to fixed stop" is possible for axes and spindles.
- "Travel to fixed stop" is possible simultaneously for several axes and parallel to the traversing of other axes.
- Multi-channel block search can be performed with calculation of all the required additional data (SERUPRO).
- Simulated axis traversal in conjunction with "travel to fixed stop" and "torque reduction" possible.
- "Vertical" axes can also be moved with FXS to a fixed stop.
6.2 Detailed description

6.2.1 Programming

Function

Travel to fixed stop

The "Travel to fixed stop" function is controlled via the FXS, FXST and FXSW commands. The activation can also be performed without traversing motion of the relevant axis. The torque is immediately limited. The fixed stop is monitored as soon as the axis is traversed.

Note

Synchronized actions

The "Travel to fixed stop" function can also be controlled via synchronized actions.

References:

Function Manual, Synchronized Actions

Travel with limited torque/force

Travel with limited torque/force can be controlled via the FOCON, FOCOF and FOC commands (see Section "Travel with limited torque/force FOC (Page 311)").

Syntax

FXS[<axis>] = <request>
FXST[<axis>] = <clamping torque>
FXSW[<axis>] = <window width>

Meaning

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FXS:</td>
<td>&quot;Travel to fixed stop&quot; function, effectiveness: Modal</td>
</tr>
<tr>
<td>&lt;Request&gt;:</td>
<td>0 = switch off through</td>
</tr>
<tr>
<td></td>
<td>1 = switch on</td>
</tr>
<tr>
<td>FXST:</td>
<td>Set clamping torque</td>
</tr>
<tr>
<td>&lt;Clamping torque&gt;:</td>
<td>Clamping torque in % of the maximum drive torque.</td>
</tr>
<tr>
<td></td>
<td>SINAMICS S120: p2003</td>
</tr>
<tr>
<td>FXSW:</td>
<td>Set monitoring window</td>
</tr>
<tr>
<td>&lt;Window width&gt;:</td>
<td>Width of the tolerance window around the fixed stop</td>
</tr>
<tr>
<td></td>
<td>Unit: mm, inch or degrees</td>
</tr>
<tr>
<td>&lt;axis&gt;:</td>
<td>Name of the channel axis, type: AXIS</td>
</tr>
</tbody>
</table>
Changes to the torque limiting (FXST)

The torque limit value can be changed in each block. The change becomes effective before executing the traversing motion programmed in the block. The torque limitation acts in addition to the acceleration limitation (ACC).

Ramp-shaped change

A time can be set using the machine data, within which the torque limit value is linearly changed.

MD37012 $MA_FIXED_STOP_TORQUE_RAMP_TIME (time until the new torque limit is reached)

Changes to the monitoring window (FXSW)

The monitoring window can be changed in each block. The change becomes effective before executing the traversing motion programmed in the block.

When the monitoring window is changed, not only does the window width change, but also the reference point of the window to the actual axis position.

"Travel to fixed stop" when the continuous-path mode is active (G64)

The following machine data can be used to set that with the selection of the (FXS) function during active continuous-path mode (G64), no exact stop is triggered at the block change (G60):

MD37060 $MA_FIXED_STOP_ACKN_MASK (monitoring PLC acknowledgements for travel to fixed stop)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Start of the traversing motion without acknowledgement by the PLC</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>Start of the traversing motion after acknowledgement by the PLC</td>
</tr>
</tbody>
</table>

Supplementary conditions

- The traversing motion to the fixed stop can be programmed as a path- or block-related or modal positioning axis motion.
- Travel to fixed stop can be selected for several machine axes simultaneously.
- For a machine axis, which is traversed to a fixed stop, it is not permissible that transformation, coupling and frame functions are active:
- The travel path and the activation of the function must be programmed in one block in the part program.
- If "Travel to fixed stop" is activated via synchronized actions, the travel path and the activation of the function can be programmed in separate blocks.
6.2 Detailed description

6.2.2 Functional sequence

6.2.2.1 Selection

Procedure

The NC detects that the function "Travel to fixed stop" is selected via the command \( \text{FXS}[x]=1 \) and signals the PLC using the IS DB31, … DBX62.4 ("Activate travel to fixed stop") that the function has been selected.

If the machine data:

\[ \text{MD37060 $MA\_FIXED\_STOP\_ACKN\_MASK} \]

is set correspondingly, the system waits for the acknowledgement of the PLC using the IS DB31, … DBX3.1 ("Enable travel to fixed stop").

The programmed target position is then approached from the start position at the programmed velocity. The fixed stop must be located between the start and target positions of the axis/spindle. A programmed torque limit (clamping torque specified via \( \text{FXST}[<axis>] \)) is effective from the start of the block, i.e. the fixed stop is also approached with reduced torque. Allowance for this limitation is made in the NC through an automatic reduction in the acceleration rate.

If no torque has been programmed in the block or since the start of the program, then the value is valid in the axis-specific machine data:

\[ \text{MD37010 $MA\_FIXED\_STOP\_TORQUE\_DEF} \]

is entered.
6.2.2.2 Fixed stop is reached

Detecting the fixed stop

Detecting the fixed stop or identifying that the machine axis has reached the fixed stop can be set using the following machine data:

MD37040 $MA_FIXED_STOP_BY_SENSOR = <value> (fixed stop detection via sensor)

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The fixed stop has been reached when the contour deviation of the machine axis has exceeded the value set in the machine data: MD37030 $MA_FIXED_STOP_THRESHOLD (threshold for fixed stop detection)</td>
</tr>
<tr>
<td>1</td>
<td>An external sensor detects when the fixed stop has been reached and informs the control via the following axial NC/PLC interface signal: DB31, ... DBX1.2 == 1 (sensor for fixed stop)</td>
</tr>
<tr>
<td>2</td>
<td>The fixed stop has been reached, if one of the conditions, specified under &lt;value&gt; == 0 OR &lt;value&gt; == 1 applies.</td>
</tr>
</tbody>
</table>

Ineffective NC/PLC interface signals

When the axis is in the "Fixed stop reached" state, the following NC/PLC interface signals have no effect:

- DB31, ... DBX1.3 (axis/spindle disable)
- DB31, ... DBX2.1 (controller enable)

Actions when the fixed stop is reached

The following actions are executed when the fixed stop is reached:

- The torque in the drive is increased up to the programmed clamping torque (FXST)
- The remaining distance to go is deleted
- The position setpoint is tracked
- The NC/PLC interface signal is set: DB31, ... DBX62.5 = 1 ("fixed stop reached")
- Executing a block change:
  - Executing a block change when the fixed stop is reached: MD37060 $MA_FIXED_STOP_ACKN_MASK, Bit 1 = 0
    The block change is immediately executed after reaching the fixed stop.
  - Block change is only executed after the acknowledgment by the PLC user program: MD37060 $MA_FIXED_STOP_ACKN_MASK, Bit 1 = 1
    After reaching the fixed stop, the block is not changed until acknowledgment by the PLC user program:
    DB31, ... DBX1.1 == 1 (acknowledge fixed stop reached)

In order to also maintain the clamping torque after the block change, the NC still outputs a setpoint for the machine axis.

- Activating the fixed stop monitoring or the monitoring window
Monitoring window

If, in the traversing block to the fixed stop or since the beginning of the program, no specific value for the monitoring window is programmed with FXSW then the value set in the machine data is active:

MD37020 $MA_FIXED_STOP_WINDOW_DEF (default for fixed stop monitoring window)

If the axis leaves the position it was in when the fixed stop was detected by more than the specified window, then alarm 20093 "Fixed stop monitoring has responded" is displayed and the "Travel to fixed stop" function is deselected.

The monitoring window must be selected by the user such that the alarm is activated only when the axis leaves the fixed stop position.

**NOTICE**

"Fixed stop reached" and when the fixed stop breaks

As soon as the "fixed stop reached" state is identified, then a speed setpoint derived from the axis-specific machine data servo gain factor (K_v) (MD32200) and the threshold for the fixed stop detection (MD37030) is entered for the drive. If the fixed stop breaks in this state, then the axis accelerates until it reaches the monitoring window limit. The velocity that is then reached is proportional to the values set in the specified machine data. For appropriately high values, it is possible that the drive accelerates up to the maximum motor speed.

Overview

![Diagram of fixed stop monitoring](image-url)

Figure 6-2  Fixed stop is reached
6.2.2.3 Fixed stop is not reached

Alarm suppression

Alarms for various causes of breakage can be suppressed using the machine data:

MD37050 $MA_FIXED_STOP_ALARM_MASK = <value>

<table>
<thead>
<tr>
<th>Value</th>
<th>Description: Suppressed alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm 20091 &quot;Fixed stop not reached&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Alarm 20091 &quot;Fixed stop not reached&quot;</td>
</tr>
<tr>
<td></td>
<td>Alarm 20094 &quot;Fixed stop aborted&quot;</td>
</tr>
<tr>
<td>3</td>
<td>Alarm 20094 &quot;Fixed stop aborted&quot;</td>
</tr>
</tbody>
</table>

Actions in the case of a fault or breakage

The following actions are executed when a fault occurs or for a breakage:

- The NC/PLC interface signal is reset: DB31, ... DBX62.4 = 0 (activate travel to fixed stop)
- Depending on the setting in the machine data, the system waits for acknowledgment:
  - MD37060 $MA_FIXED_STOP_ACKN_MASK
  - DB31, ... DBX3.1 == 0 (enable travel to fixed stop)
- Withdrawing torque limits
- Executing a block change
Figure 6-3 Fixed stop is not reached

### 6.2.2.4 Deselection

The "travel to fixed stop" function is deselected using the command $\text{FXS[<axis>] = 0}$ in a block of an NC program.

#### Actions when deselecting the function

When deselecting the function, the following actions are executed:

- A preprocessing stop is initiated ($\text{STOPRE}$)
- The NC/PLC interface signals are reset
  - $\text{DB31, ... DBX62.4 = 0}$ (activate travel to fixed stop)
  - $\text{DB31, ... DBX62.5 = 0}$ (fixed stop reached)
- Depending on the machine data, the system waits for acknowledgment from the PLC user program:
  - $\text{MD37060 $MA\_FIXED\_STOP\_ACKN\_MASK}$
  - $\text{DB31, ... DBX3.1 == 0}$ (enable travel to fixed stop)
  - $\text{DB31, ... DBX1.1 == 0}$ (acknowledge fixed stop reached)
- Exiting the follow-up mode
- Axis resumes closed-loop position control
Pulse enable

The pulse enable or pulse inhibit can be canceled via:

- Drive: Via terminal EP (enable pulses)
- NC/PLC interface signal: DB31, ... DBX21.7 ("pulse enable")

The behavior at the fixed stop can be set via the following machine data:

MD37002 $MA_FIXED_STOP_CONTROL, bit 0 and bit 1 (sequence control for traversing to fix stop)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Behavior for pulse <em>inhibit</em> at the fixed stop</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>&quot;Travel to fixed stop&quot; is aborted</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>&quot;Travel to fixed stop&quot; is interrupted, i.e. the drive is without power.</td>
</tr>
<tr>
<td>1</td>
<td>Behavior for pulse <em>enable</em> at the fixed stop</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>The torque is injected suddenly.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>The torque is linearly ramped-up over the time specified in the machine data: MD37012 $MA_FIXED_STOP_TORQUE_RAMP_TIME</td>
</tr>
</tbody>
</table>

**Special case: Deleting the pulse enable during deselection**

If, during deselection, the function in state: "Waiting for PLC acknowledgements" the pulse enable is deleted, the torque limit is reduced to 0. In this phase, if pulse enable is set again torque is no longer established in the drive. Once the deselection has been completed the axis can be traversed normally again.
Overview

![Graph](image)

Figure 6-4  Fixed stop deselection

6.2.3 Behavior during block search

**Function**

**Block search with calculation**
- If the target block is located in a program section in which the axis must stop at a fixed limit, then the fixed stop is approached if it has not yet been reached.
- If the target block is located in the program section in which the axis must not stop at a fixed limit, then the axis leaves the fixed stop if it is still positioned there.
- If the axis is in the "Fixed stop reached" state, message 10208 "Press NC start to continue the program" is displayed. The program can be continued with NC start.
- Clamping torque $FXST$ and monitoring window $FXSW$ have the value that they have for normal program processing at the start of the target block.

**Block search without calculation**

The commands $FXS$, $FXST$ and $FXSW$ are ignored during the block search.
Effectiveness of FOCON/FOCOF

The state of the modal-acting torque/force reduction FOCON/FOCOF is maintained during the search and is effective in the approach block.

Block search with FXS or FOC

The user selects FXS or FOC in a program area of the target block in order to acquire all states and functions of the machining last valid. The NC starts the selected program in Program test mode automatically. After the target block has been found, the NC stops at the beginning of the target block, deselects Program test internally again and displays the Stop condition “Search target found” in its block display.

⚠️ CAUTION

SERUPRO approach does not really take the FXS command into account.
The approach to the programmed end position of the FXS block is only simulated without torque limitation.

If FXS is located between the beginning of the program and the search target, the command is not executed by the NC. The motion is only simulated up to the programmed end point.
The user can log the turning on and turning off of FXS in the part program. If necessary, the user can start an ASUB in order to activate or deactivate FXS in this SERUPRO-ASUP.

System variable

The reference and actual state of the “travel to fixed stop” function can be read using the following system variable:
- $AA_FXS (reference state)
- $VA_FXS (actual state)

SERUPRO: $AA_FXS (reference state)

During SERUPRO, $AA_FXS supplies the following values depending on the activation status of the “Travel to fixed stop” function:

<table>
<thead>
<tr>
<th>Activation status of the &quot;travel to fixed stop&quot; function</th>
<th>System variable $AA_FXS ==</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Deactivated&quot;</td>
<td>0 (axis not at fixed stop)</td>
</tr>
<tr>
<td>&quot;Activated&quot;</td>
<td>3 (&quot;Travel to fixed stop&quot; selection is active)</td>
</tr>
</tbody>
</table>

Note

During SERUPRO, the system variable $AA_FXS only supplies the values 0 and 3. As a result, based on $AA_FXS, the program sequence can be changed with SERUPRO compared to the normal program execution for program branches.

SERUPRO: $VA_FXS (actual state)

During SERUPRO, the variable $VA_FXS always supplies the real state of axis on the machine.
Example
The current state of the "Travel to fixed stop" function can be determined in the SERUPRO ASUB via the system variables $AA\_FXS$ and $VA\_FXS$, and the appropriate response initiated:

<table>
<thead>
<tr>
<th>Program code: FXS_SERUPRO_ASUP.MPF</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 WHEN ($AA_FXS[\text{X}]$==3) AND ($VA_FXS[\text{X}]$==0) DO</td>
<td></td>
</tr>
<tr>
<td>FXS[\text{X}]$=1</td>
<td></td>
</tr>
<tr>
<td>; Reference==&quot;Selection not active&quot; AND</td>
<td></td>
</tr>
<tr>
<td>; Actual==&quot;Axis not at fixed stop&quot;</td>
<td></td>
</tr>
<tr>
<td>; \text{-&gt; }&quot;Switch on&quot;</td>
<td></td>
</tr>
<tr>
<td>N200 WHEN ($AA_FXS[\text{X}]$==0) AND ($VA_FXS[\text{X}]$==1) DO</td>
<td></td>
</tr>
<tr>
<td>FXS[\text{X}]$=0</td>
<td></td>
</tr>
<tr>
<td>; Reference==&quot;Axis not at fixed stop&quot; AND</td>
<td></td>
</tr>
<tr>
<td>; Actual==&quot;Fixed stop successfully approached&quot;</td>
<td></td>
</tr>
<tr>
<td>; \text{-&gt; }&quot;Switch on&quot;</td>
<td></td>
</tr>
<tr>
<td>N1020 REPOSA</td>
<td></td>
</tr>
<tr>
<td>; Reapproaching the contour linearly with all axes</td>
<td></td>
</tr>
</tbody>
</table>

Displaying the REPOS offset

Once the search target has been found, for each axis, the actual state regarding the "Travel to fixed stop" is displayed using the following NC/PLC interface signals:

- DB31, ... DBX62.4 (activate travel to fixed stop)
- DB31, ... DBX62.5 (fixed stop reached)

Example:
If the axis is at the fixed stop and the target block is available after deselection of FXS, the new target position is displayed via DB31, ... DBX62.5 (fixed stop reached) as the REPOS offset.

REPOS and FXS

With REPOS, the functionality of FXS is repeated automatically and called FXS-REPOS in the following. This sequence is comparable to the FXS\_SERUPRO\_ASUP.MPF program. Whereby every axis is taken into account and the last torque programmed before the search target is used.

The user can treat FXS separately in a SERUPRO ASUB.

The following then applies:

Every FXS action executed in the SERUPRO ASUB automatically takes care of

$AA\_FXS[<\text{axis}>] = VA\_FXS[<\text{axis}>]$.

This deactivates FXS-REPOS for axis X.
Deactivating FXS-REPOS

FXS-REPOS is deactivated by:

- An FXS synchronized action which refers to REPOSA
- $AA_{FXS}[X] = $VA_{FXS}[X]$ in the SERUPRO_ASUB

Note

A SERUPRO ASUB without FXS treatment or no SERUPRO ASUB results automatically in FXS-REPOS.

CAUTION

**Speed too high for FXS-REPOS**

FXS-REPOS traverse all path axes together to the target position. Axes with and without FXS treatment thus traverse together with the G code and feedrate valid in the target block. As a result, the fixed stop may be approached in rapid traverse ($G0$) or higher velocity.

FOC-REPOS

FOC-REPOS behaves in the same way as FXS-REPOS.

A changing torque characteristic during the program preprocessing cannot be implemented with FOC-REPOS.

Example

Axis X is traversed from position 0 to 100. FOC is switched on every 20 millimeters for 10 millimeters. The resulting torque characteristic is generated with non-modal FOC and cannot be traced by FOC-REPOS. Axis X is traversed by FOC-REPOS with or without FOC in accordance with the last programming before the target block.

For programming examples of FXS “Travel to fixed stop”, see Section "Program test (Page 471)".

6.2.4 Behavior for reset and function abort

NC reset

As long as the function is still not in the “Successful travel to fixed stop” state, the travel to fixed stop can be aborted with NC reset.

Even when the fixed stop has already been approached, but the specified stop torque not yet fully reached, then the function can still be aborted with NC reset. The position setpoint of the axis is synchronized with the current actual position.

As soon as the function is in the "Successful travel to fixed stop" state, the function also remains active after the NC reset.
6.2 Detailed description

Function abort

A function abort can be triggered by the following events:

- Emergency stop

<table>
<thead>
<tr>
<th>CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dangerous machine situations possible for travel to limit stop</strong></td>
</tr>
</tbody>
</table>

It must be ensured that no dangerous machine situations occur while travel to fixed stop is active when an "Emergency stop" is triggered or reset.

For example, behavior when setting and canceling the pulse enable:

MD37002 $MA_FIXED_STOP_CONTROL, bit 0 (behavior for pulse disable at the stop)

- Bit 0 = 0: Travel to fixed stop is aborted
- Bit 0 = 1: Travel to fixed stop is interrupted, i.e. the drive is without power

Once the pulse disable is canceled again, the drive presses with the stop torque again.

Note

NC and drive have no power during "Emergency stop", i.e. the PLC must react.

- Functional state: "Fixed stop not reached"
- Functional state: "Fixed stop aborted"
- Aborted by the PLC user program:
  DB31, ... DBX62.4 = 0 ("Activate travel to fixed stop")
- Cancellation of the pulse enable and machine data parameterization:
  MD37002 $MA_FIXED_STOP_CONTROL, bit 0 = 0 (see above)

6.2.5 Behavior with regard to other functions

**Measurement with delete distance-to-go**

"Travel to fixed stop" (FXS) cannot be programmed in a block with "Measurement with delete distance-to-go" (MEAS). Except when one function acts on a path axis while the other acts on a positioning axis or both functions act on positioning axes.

**Contour monitoring**

The axis contour monitoring function is inoperative while "Travel to fixed stop" is active.

**Positioning axes**

For "Travel to fixed stop" with positioning axes POSA, the block change is executed even when the positioning axis has not yet reached the fixed stop by this time.

**Vertical axes**

The "Travel to fixed stop" function can be used for vertical axes even when alarms are active.

If a function-specific alarm occurs for a vertical axis when traveling to fixed stop, the NC/PLC interface signal DB11, DBX6.3 (mode group ready) is not reset: This means that the corresponding drive is not de-energized.
This corresponds to an electronic weight compensation for the vertical axis and can be configured via the following machine data:

**MD37052 $MA_FIXED_STOP_ALARM_REACTION**

**References**
Further information on vertical axes can be found in:

- SINAMICS S120 Function Manual
- Function Manual, Extended Functions; Compensation (K3), Section: Electronic counterweight

### 6.2.6 Setting data

The values programmed via the function-specific **FXS, FXST** and **FXSW** commands are written block-synchronously to the following, immediately effective, axis-specific setting data:

**Switching the function on/off**

**SD43500 $SA_FIXED_STOP_SWITCH** (selection/deselection of travel to fixed stop)

**Clamping torque**

**SD43510 $SA_FIXED_STOP_TORQUE** (clamping torque)

---

**Note**

**Clamping torque greater than 100%**

A value for the clamping torque in **SD43510** greater than 100% of the maximum motor torque is only advisable for a short time. In addition, the maximum motor torque is limited by the drive. For example, the following drive parameters have a limiting effect:

- **p1520/p1521 upper torque limit/force limit / lower torque limit/force limit**
- **p1522/p1523 upper torque limit/force limit / lower torque limit/force limit**
- **p1530/p1531 power limit, motoring / power limit, regenerating**
- **p0640 current limit**
- **p0326 motor stall force correction factor**

Detailed information on the drive parameters and the functions can be found in:

**References**

- SINAMICS S120/S150 List Manual
- SINAMICS S120 Function Manual

**Monitoring window**

**SD43520 $SA_FIXED_STOP_WINDOW** (monitoring window)
Default setting

The defaults for the setting data are set via the following machine data:

- Clamping torque:
  MD37010 $MA_FIXED_STOP_TORQUE_DEF (default clamping torque)
- Monitoring window:
  MD37020 $MA_FIXED_STOP_WINDOW_DEF (default monitoring window)

Effectiveness

The setting data for the clamping torque and monitoring window takes effect immediately. In this way, the clamping state can be adapted to the machining situation at any time by the operator or via the PLC user program.

References

Further detailed information on the machine and setting data can be found in:

List Manual, Detailed Machine Data Description

See also

Z1: NC/PLC interface signals (Page 1551)

6.2.7 System variables

Reference/actual state

The reference and actual state of the "travel to fixed stop" function can be read using the following system variables:

- $AA_FXS = <value> (status, "Travel to fixed stop" reference state)
- $VA_FXS = <value> (status, "Travel to fixed stop" actual state)

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Axis is not at fixed stop</td>
</tr>
<tr>
<td>1</td>
<td>Fixed stop was successfully approached</td>
</tr>
<tr>
<td>2</td>
<td>Approach to fixed stop failed</td>
</tr>
<tr>
<td>3</td>
<td>Selection &quot;Travel to fixed stop&quot; is active.</td>
</tr>
<tr>
<td>4</td>
<td>Fixed stop was detected.</td>
</tr>
<tr>
<td>5</td>
<td>Travel to fixed stop deselection active</td>
</tr>
</tbody>
</table>
Additional information

If errors occurred when traversing to the fixed stop ($VA_{FXS} == 2$), then additional information is displayed in the following system variables:

- $VA_{FXS}_{INFO} = \text{<value>} \text{ (additional information when "traveling to fixed stop")}$

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No additional information available</td>
</tr>
<tr>
<td>1</td>
<td>No approach motion programmed</td>
</tr>
<tr>
<td>2</td>
<td>Programmed end position reached, motion stopped</td>
</tr>
<tr>
<td>3</td>
<td>Cancellation using a reset, $DB21, \ldots DBX7.7 == 1$</td>
</tr>
<tr>
<td>4</td>
<td>Monitoring window was exited</td>
</tr>
<tr>
<td>5</td>
<td>Torque reduction rejected by drive</td>
</tr>
<tr>
<td>6</td>
<td>PLC has withdrawn the enable, $DB31, \ldots DBX3.1 == 0 \text{ (travel to fixed stop enabled) }$</td>
</tr>
</tbody>
</table>

Application example for $AA_{FXS}$

In order that a block change is executed, no alarm should be triggered when a fault occurs. The cause is then determined using the system variable $VA_{FXS}_{INFO}$ and a specific response initiated.

Requirement: MD37050 $MA_{FIXED\_STOP\_ALARM\_MASK} = 0 \text{ (when traveling to fixed stop, no alarm initiated) }$

Program code

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>X300 Y500 F200 FXS[X1]=1 FXST[X1]=25 FXSN[X1]=5</td>
<td>; Travel to fixed stop</td>
</tr>
<tr>
<td>IF $AA_{FXS}[X1] == 2$ GOTOF FXS_ERROR</td>
<td>; IF fixed stop == reached</td>
</tr>
<tr>
<td>R100=$AA_{IM}[X1]$</td>
<td>; THEN (normal case)</td>
</tr>
<tr>
<td>IF R100 ...</td>
<td>; Evaluation of the</td>
</tr>
<tr>
<td>...</td>
<td>; actual position</td>
</tr>
<tr>
<td>GOTOF PROG_END</td>
<td>;</td>
</tr>
<tr>
<td>FXS_ERROR:</td>
<td>; ELSE (error case)</td>
</tr>
<tr>
<td>CASE($VA_{FXS}_{INFO}[X1]) OF 0 GOTOF LABEL_0 OF 1 GOTOF LABEL_1 ...</td>
<td>; Error handling</td>
</tr>
<tr>
<td>LABEL_0: ...</td>
<td></td>
</tr>
<tr>
<td>GOTOF CASE_END</td>
<td></td>
</tr>
<tr>
<td>LABEL_1: ...</td>
<td></td>
</tr>
<tr>
<td>GOTOF CASE_END</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>CASE END:</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>PROG_END: M30</td>
<td>; ENDF end of program</td>
</tr>
</tbody>
</table>
6.2.8 Alarms

Alarm 20091 "Fixed stop not reached"
If the fixed stop position is not reached during travel to fixed stop, alarm 20091 "Fixed stop not reached" is displayed and a block change executed.

Alarm 20092 "Travel to fixed stop is still active"
If there is a travel request or renewed function selection for the axis after the fixed stop has been reached, alarm 20092 "Travel to fixed stop is still active" is displayed.

Alarm 20093 "Standstill monitoring at fixed stop has triggered"
If an axis has reached the fixed stop and is then moved out of this position by more than the value specified in the setting data SD43520 FIXED_STOP_WINDOW (fixed stop monitoring window) alarm 20093 "Standstill monitoring at fixed stop has triggered" is displayed, travel to fixed stop for this axis is deselected and the following system variable set:

$AA_FXS[x] = 2

Alarm 20094 "Function has been aborted"
The travel to fixed stop is aborted if the clamping torque can no longer be applied due to the cancellation of the pulse enable, or the requested acknowledgement signal at the NC/PLC interface has been reset:

- Acknowledgement signal required: MD37060 $MA_FIXED_STOP_ACKN_MASK, bit 0 = 1
- Acknowledgement signal: DB31, ... DBX3.1 == 0 (enable travel to fixed stop)

Enabling the fixed stop alarms
The following machine data can be use to set whether the fixed stop alarms

- Alarm 20091 "Fixed stop not reached",
- Alarm 20094 "Fixed stop aborted"

are displayed:

MD37050 $MA_FIXED_STOP_ALARM_MASK (enable of the fixed stop alarms)

Settable functional behavior for fixed stop alarms
The following machine data can be used to set that the function is not aborted when function-specific alarms occur:

- Alarm 20090 Travel to fixed stop not possible
- Alarm 20091 Fixed stop not reached
- Alarm 20092 Travel to fixed stop is still active
- Alarm 20093 Standstill monitoring at fixed stop has triggered
- Alarm 20094 Travel to fixed stop aborted

MD37052 $MA_FIXED_STOP_ALARM_REACTION (reaction to fixed stop alarms)
Alarm suppression after new programming

Travel to fixed stop can be used for simple measuring processes.

For example, it is possible to carry out a check for tool breakage by measuring the tool length by traversing onto a defined obstacle. To do so, the fixed stop alarm must be suppressed. When the function for clamping workpieces is then used "normally," the alarm can be activated using part program commands.

6.2.9 Travel with limited torque/force FOC

Function

Using the "Force Control" function to traverse with limited torque/force, when traversing, the maximum permissible torque/force can be limited to a percentage of the maximum possible axis torque. The limit value can be changed at any time, also while the axis is traversing. The changes can be realized in the interpolator clock cycle, dependent on distance, time or on any other variable. The "Force Control" function can also be programmed in synchronized actions. The function can be permanently activated, or block related.

Programming

Syntax

FOCON[<axis>]
FOCOF[<axis>]
FOC[<axis>]

Meaning

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOCON:</td>
<td>Activate torque/force limiting</td>
</tr>
<tr>
<td>FOCOF:</td>
<td>Deactivate torque/force limiting</td>
</tr>
<tr>
<td>FOC:</td>
<td>Activate non-modal torque/force limiting</td>
</tr>
<tr>
<td>&lt;axis&gt;:</td>
<td>Channel axis name, type: AXIS</td>
</tr>
</tbody>
</table>

Example

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 FOCON[X]</td>
<td>Modal activation of the torque limit</td>
</tr>
<tr>
<td>N20 X100 Y200 FXST[X]=15</td>
<td>X travels with reduced torque (15%)</td>
</tr>
<tr>
<td>N30 FXST[X]=75 X20</td>
<td>Changing the torque to 75%,</td>
</tr>
<tr>
<td>N40 FOCOF[X]</td>
<td>Disable torque limit</td>
</tr>
</tbody>
</table>
Parameterization

Machine data

- MD37010 $MA_FIXED_STOP_TORQUE_DEF (default for fixed stop clamping torque)
  The value specified in the machine data is effective after activating the function, as long as no explicit value is programmed using FXST.
- MD36042 $MA_FOC_STANDSTILL_DELAY_TIME (delay time standstill monitoring for active torque/force limiting)

Permanent activation (FOCON/FOCOF)

Permanent activation of the "Force Control" function after POWER ON or RESET can be set using the machine data:

MD37080 $MA_FOC_ACTIVATION_MODE, bit 0 and bit 1 (basic setting of the modal force/torque limiting)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>The &quot;Force Control&quot; function is not active after POWER ON.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>The &quot;Force Control&quot; function is active after POWER ON.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>The &quot;Force Control&quot; function is not active after RESET.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>The &quot;Force Control&quot; function is active after RESET.</td>
</tr>
</tbody>
</table>

If a limiting torque or force is programmed using FXST = <value> before activation, then this is effective from activation onwards.

If a limiting torque or force was not programmed using FXST = <value> before activation, then the value specified in the machine data is effective as standard.

MD37010 $MA_FIXED_STOP_TORQUE_DEF (default for fixed stop clamping torque)

Block-related activation (FOC)

The function for the actual block is activated using the FOC command.

Activating the function using a synchronized action takes effect up to the end of the current part program block.

Priority of FXS before FOC

Activating the "Travel to fixed stop" function FXS has a higher priority than the "Force Control" function. If "Travel to fixed stop" is active at the same time as "Force Control", then the first is executed.

Deselecting the "Travel to fixed stop" function FXS cancels the clamping. A "Force Control" function that is at same time permanently active, remains active.

System variable

Status of the "Force Control" function

The status of the "Force Control" function can be read using system variable $AA_FOC.
Changing the status of the "Travel to fixed stop" function \texttt{FXS} does not change the status of the "Force Control" function.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{Value} & \textbf{Meaning} \\
\hline
0 & FOC not active \\
1 & FOC modal active \\
2 & FOC non-modal active \\
\hline
\end{tabular}
\end{table}

\textbf{Status of torque limiting}

The actual status of the torque limiting can be read using system variable
\texttt{$VA\_TORQUE\_AT\_LIMIT$}.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{Value} & \textbf{Meaning} \\
\hline
0 & actually effective torque < torque limit value \\
1 & actually effective torque == torque limit value \\
\hline
\end{tabular}
\end{table}

\textbf{Restrictions}

the "Force Control" function has the following restrictions:

- The change of the torque/force limitation representing itself as an acceleration limitation is \textit{only taken into account in the traversing movement at block limits} (see command \texttt{ACC}).
- Only FOC: \textbf{No monitoring} is possible from the NC/PLC interface to check that the active torque limit has been reached.
- If the acceleration limitation is not adapted accordingly, an increase in the following error occurs during the traversing motion.
- If the acceleration limitation is not adapted accordingly, the end-of-block position is possibly reached later than specified in: MD36040 \texttt{$MA\_STANDSTILL\_DELAY\_TIME$}.  
  The machine data: MD36042 \texttt{$MA\_FOC\_STANDSTILL\_DELAY\_TIME$} is introduced for this and monitored in this status.

\textbf{Possible application for link and container axes}

All axes that can be traversed in a channel, i.e. also link axes and container axes, can be traversed to fixed stop.

\textbf{References:}
Function Manual, Extended Functions; Several Operator Panels on multiple NCUs, Distributed Systems (B3)

The status of the machine axis is kept in the case of a container rotation, i.e. a clamped machine axis remains at the stop.

If a modal torque limitation has been activated with FOCON, this is kept for the machine axis even after a container rotation.
### Example 1: Travel to fixed stop with static synchronized actions

Travel to fixed stop (FXS) is initiated when requested via R parameter ($R1$) in a static synchronized action.

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>N10 IDS=1 WHENEVER</strong></td>
<td>Static synchronized action 1:</td>
</tr>
<tr>
<td>$($R1==1$) AND $($AA_FXS[Y]==0$)) DO</td>
<td>R1==1 (FXS for Y requested) AND Avoidance of multiple selection $AA_FXS[Y]==0$ (axis not at limit): $R1=0$ FXS[Y]=1 reset R1, activate FXS for Y</td>
</tr>
<tr>
<td>$R1=0$ FXS[Y]=1</td>
<td>Limit torque: 10%</td>
</tr>
<tr>
<td>FXST[Y]=10</td>
<td>Axial feedrate Y: 200</td>
</tr>
<tr>
<td>FA[Y]=200</td>
<td>Positioning movement Y</td>
</tr>
<tr>
<td>POS[Y]=150</td>
<td></td>
</tr>
<tr>
<td><strong>N11 IDS=2 WHENEVER</strong></td>
<td>Static synchronized action 2:</td>
</tr>
<tr>
<td><strong>N12 IDS=3 WHENEVER</strong></td>
<td>Static synchronized action 3:</td>
</tr>
<tr>
<td><strong>N13 IDS=4 WHENEVER</strong></td>
<td>Static synchronized action 4:</td>
</tr>
<tr>
<td>FA[Y]=1000 POS[Y]=0</td>
<td>Positioning movement Y</td>
</tr>
<tr>
<td><strong>N20 FXS[Y]=0 G0 G90 X0 Y0</strong></td>
<td>Set initial settings: FXS deselected, X and Y at initial position</td>
</tr>
<tr>
<td><strong>N30 RELEASE(Y)</strong></td>
<td>Release Y for movements in synchronized actions</td>
</tr>
<tr>
<td><strong>N60 GET(Y)</strong></td>
<td>Include axis Y back into the path group</td>
</tr>
</tbody>
</table>

**Note**

**Avoidance of multiple selection for FXS**

To avoid a multiple selection, we recommend that prior to activating FXS, query either the $AA_FXS==0$ system variable or a user-specific flag. See above, program example N10.
Example 2: Traveling to fixed stop with block-related synchronized actions

“Travel to fixed stop” is activated from a specific position of the traversing motion of the following block:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G0 G90 X0</td>
<td>; Starting position</td>
</tr>
<tr>
<td>N20 WHEN $AA_IW[X]&gt;17 DO FXS[X]=1</td>
<td>; Synchronized action: Actual position of X axis &gt; 17; -&gt; activate FXS for axis X</td>
</tr>
<tr>
<td>N30 G1 F200 X100</td>
<td>; Traversing motion for axis X</td>
</tr>
</tbody>
</table>

Note

A block-related synchronized action is processed in the following main program block.

6.4 Data lists

6.4.1 Machine data

6.4.1.1 Axis/spindle-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>36042</td>
<td>FOC_STANDSTILL_DELAY_TIME</td>
<td>Delay time, standstill monitoring for active torque/force limiting</td>
</tr>
<tr>
<td>37000</td>
<td>FIXED_STOP_MODE</td>
<td>Travel to fixed stop mode</td>
</tr>
<tr>
<td>37002</td>
<td>FIXED_STOP_CONTROL</td>
<td>Sequence monitoring for travel to fixed stop</td>
</tr>
<tr>
<td>37010</td>
<td>FIXED_STOP_TORQUE_DEF</td>
<td>Fixed stop clamping torque default setting</td>
</tr>
<tr>
<td>37012</td>
<td>FIXED_STOP_TORQUE_RAMP_TIME</td>
<td>Time until the modified torque limit is reached</td>
</tr>
<tr>
<td>37020</td>
<td>FIXED_STOP_WINDOW_DEF</td>
<td>Default for fixed stop monitoring window</td>
</tr>
<tr>
<td>37030</td>
<td>FIXED_STOP_THRESHOLD</td>
<td>Threshold for fixed stop detection</td>
</tr>
<tr>
<td>37040</td>
<td>FIXED_STOP_BY_SENSOR</td>
<td>Fixed stop detection via sensor</td>
</tr>
<tr>
<td>37050</td>
<td>FIXED_STOP_ALARM_MASK</td>
<td>Enabling the fixed stop alarms</td>
</tr>
<tr>
<td>37052</td>
<td>FIXED_STOP_ALARM_REACTION</td>
<td>Reaction to fixed stop alarms</td>
</tr>
<tr>
<td>37060</td>
<td>FIXED_STOP_ACKN_MASK</td>
<td>Monitoring of PLC acknowledgments for travel to fixed stop</td>
</tr>
<tr>
<td>37070</td>
<td>FIXED_STOP_ANA_TORQUE</td>
<td>Torque limit on fixed stop approach for analog drives</td>
</tr>
<tr>
<td>37080</td>
<td>FOC_ACTIVATION_MODE</td>
<td>Initial setting of the modal torque/force limiting</td>
</tr>
</tbody>
</table>
6.4.2 Setting data

6.4.2.1 Axis/spindle-specific setting data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $SA___</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>43500</td>
<td>FIXED_STOP_SWITCH</td>
<td>Selection of travel to fixed stop</td>
</tr>
<tr>
<td>43510</td>
<td>FIXED_STOP_WINDOW</td>
<td>Fixed stop clamping torque</td>
</tr>
<tr>
<td>43520</td>
<td>FIXED_STOP_TORQUE</td>
<td>Fixed stop monitoring window</td>
</tr>
</tbody>
</table>

6.4.3 Signals

6.4.3.1 Signals to axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledge fixed stop reached</td>
<td>DB31, ... .DBX1.1</td>
<td>DB380x.DBX1.1</td>
</tr>
<tr>
<td>Sensor for fixed stop</td>
<td>DB31, ... .DBX1.2</td>
<td>DB380x.DBX1.2</td>
</tr>
<tr>
<td>Axis/spindle disable</td>
<td>DB31, ... .DBX1.3</td>
<td>DB380x.DBX1.3</td>
</tr>
<tr>
<td>Controller enable</td>
<td>DB31, ... .DBX2.1</td>
<td>DB380x.DBX2.1</td>
</tr>
<tr>
<td>Travel to fixed stop enabled</td>
<td>DB31, ... .DBX3.1</td>
<td>DB380x.DBX3.1</td>
</tr>
</tbody>
</table>

6.4.3.2 Signals from axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate travel to fixed stop</td>
<td>DB31, ... .DBX62.4</td>
<td>DB390x.DBX2.4</td>
</tr>
<tr>
<td>Fixed stop reached</td>
<td>DB31, ... .DBX62.5</td>
<td>DB390x.DBX2.5</td>
</tr>
</tbody>
</table>
G2: Velocities, setpoint / actual value systems, closed-loop control

7.1 Brief description

The description of functions explains how to parameterize a machine axis in relation to:

- Actual-value/measuring systems
- Setpoint system
- Operating accuracy
- Travel ranges
- Axis velocities
- Control parameters

7.2 Velocities, traversing ranges, accuracies

7.2.1 Velocities

Maximum path and axis velocities and spindle speed

The maximum path and axis velocities and spindle speed are influenced by the machine design, the dynamic response of the drive and the limit frequency of the actual-value acquisition (encoder limit frequency).

The maximum axis velocity is defined in machine data:

MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

The maximum permissible spindle speed is specified using machine data:

MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed)

For explanations, see Section "S1: Spindles (Page 1199)".

With a high feedrate (resulting from programmed feedrates and feedrate override), the maximum path velocity is limited to $V_{\text{max}}$.

This automatic feedrate limiting can lead to a drop in velocity over several blocks with programs generated by CAD systems with extremely short blocks.
Example:

\[ \text{Interpolation cycle} = 12 \text{ ms} \]

\[ \text{N10 G0 X0 Y0; [mm]} \]

\[ \text{N20 G0 X100 Y100; [mm]} \]

\[ \Rightarrow \text{Path length programmed in block} = 141.42 \text{ mm} \]

\[ \Rightarrow V_{\text{max}} = (141.42 \text{ mm/12 ms}) \times 0.9 = 10606.6 \text{ mm/s} = 636.39 \text{ m/min} \]

Minimum path, axis velocity

The following restriction applies to the minimum path or axis velocity:

\[ V_{\text{min}} \geq \frac{10^{-3}}{\text{Computational resolution} \times \text{interp. cycle [s]}} \]

The computational resolution is defined using machine data:

MD10200 $\text{MN\_INT\_INCR\_PER\_MM}$ (computational resolution for linear positions)

or

MD10210 $\text{MN\_INT\_INCR\_PER\_DEG}$ (computational resolution for angular positions)

If \( V_{\text{min}} \) is not reached, no traversing is carried out.

Example:

MD10200 $\text{MN\_INT\_INCR\_PER\_MM} = 1000 \text{ [incr./mm]};$

Interpolation cycle = 12 ms;

\[ \Rightarrow V_{\text{min}} = 10^{-3}/(1000 \text{ incr/mm} \times 12 \text{ ms}) = 0.005 \text{ mm/min}; \]

The value range of the feedrates depends on the computational resolution selected.

For the standard assignment of machine data:

MD10200 $\text{MN\_INT\_INCR\_PER\_MM}$

(computational resolution for linear positions) (1000 incr./mm)

or

MD10210 $\text{MN\_INT\_INCR\_PER\_DEG}$

(computational resolution for angular positions) (1000 incr./deg.)

The following value range can be programmed with the specified resolution:

<table>
<thead>
<tr>
<th>Range of values for path feed F and geometry axes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric system:</td>
</tr>
<tr>
<td>0.001 ≤ F ≤ 999,999.999 [mm/min, mm/rev, degrees/min, degrees/rev]</td>
</tr>
</tbody>
</table>
Range of values for feedrate for positioning axes:

<table>
<thead>
<tr>
<th>Metric system:</th>
<th>Inch system:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.001 ≤ FA ≤ 999,999.999 [mm/min, mm/rev, degrees/min, degrees/rev]</td>
<td>0.001 ≤ FA ≤ 399,999.999 [inch/min, inch/rev]</td>
</tr>
</tbody>
</table>

Range of values for spindle speed S:

0.001 ≤ S ≤ 999,999.999 [rpm]

If the computational resolution is increased/decreased by a factor, then the value ranges change accordingly.

### 7.2.2 Traversing ranges

Range of values of the traversing ranges

The range of values of the traversing range depends on the computational resolution selected.

For the standard assignment of machine data:

MD10200 $MN_INT_INCR_PER_MM

(computational resolution for linear positions) (1000 incr./mm)

or

MD10210 $MN_INT_INCR_PER_DEG

(computational resolution for angular positions) (1000 incr./deg.)

The following value range can be programmed with the specified resolution:

<table>
<thead>
<tr>
<th></th>
<th>G71 [mm, degrees]</th>
<th>G70 [inch, degrees]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>Range</td>
<td></td>
</tr>
<tr>
<td>Linear axes X, Y, Z, etc.</td>
<td>± 999,999.999</td>
<td>± 399,999.999</td>
</tr>
<tr>
<td>Rotary axes A, B, C, etc.</td>
<td>± 999,999.999</td>
<td>± 999,999.999</td>
</tr>
<tr>
<td>Interpolation parameters I, J, K</td>
<td>± 999,999.999</td>
<td>± 399,999.999</td>
</tr>
</tbody>
</table>

The unit of measurement of rotary axes is always degrees.

If the computational resolution is increased/decreased by a factor of 10, the ranges of values change accordingly.

The traversing range can be limited by software limit switches and working areas (see Section "A3: Axis Monitoring, Protection Zones (Page 79)").

For special features for a large traversing range for linear and rotary axes, see Section "R1: Referencing (Page 1149)"

The traversing range for rotary axes can be limited via machine data.
7.2.3 Positioning accuracy of the control system

**Actual-value resolution and computational resolution**

The positioning accuracy of the control depends on the actual-value resolution (=encoder increments/(mm or degrees)) and the computational resolution (=internal increments/(mm or degrees)).

The coarse resolution of these two values determines the positioning accuracy of the control.

The input resolution, interpolator and position-control cycle selections have no effect on this accuracy.

As well as limiting using MD32000, the control limits the maximum path velocity in relation to the situation and according to the following formula:

\[
\text{Internal increments / mm} = \frac{1}{\text{Encoder increments / mm} \times \text{ENC_RESOL}[n] \times \text{ENC_PULSE_MULT}[n]}
\]

\[
\times \frac{\text{DRIVE_ENC_RATIO_NUMERA}[n]}{\text{DRIVE_ENC_RATIO_DENOM}[n]}
\]

\[
\times \frac{\text{DRIVE_AX_RATIO_DENOM}[n]}{\text{DRIVE_AX_RATIO_NUMERA}[n]}
\]

\[
\times \text{LEADSCREW_PITCH}
\]

\[
\times \text{INT_INCR_PER_MM}
\]

7.2.4 Input/display resolution, computational resolution

**Resolutions: Differences**

Resolutions, e.g. resolutions of linear and angular positions, velocities, accelerations and jerk, must be differentiated as follows:

- **Input resolution**
  Data is input via the control panel or part programs.

- **Display resolution**
  Data is displayed via the control panel.

- **Computational resolution**
  Data input via the control panel or part program is displayed internally.
The input and display resolution is determined by the specified operator panel front used, whereby the display resolution for position values with the machine data:

MD9004 $MM_DISPLAY_RESOLUTION (display resolution)
can be changed.

The machine data:

MD9011 $MM_DISPLAY_RESOLUTION_INCH (display resolution for INCH measuring system)
can be used to configure the display resolution for position values with inch setting. This allows you to display up to six decimal places with the inch setting.

For the programming of part programs, the input resolutions listed in the Programming Guide apply.

The desired computational resolution is defined using the machine data:

MD10200 $MN_INT_INCR_PER_MM (computational resolution for linear positions)
and

MD10210 $MN_INT_INCR_PER_DEG (computational resolution for angular positions).

It is independent of the input/display resolution but should have at least the same resolution.

The maximum number of places after the decimal point for position values, velocities, etc., in the part program and the number of places after the decimal point for tool offsets, zero offsets, etc. (and therefore also for the maximum possible accuracy) is defined by the computational resolution.

The accuracy of angle and linear positions is limited to the computational resolution by rounding the product of the programmed value with the computational resolution to an integer number.

To make the rounding clear, powers of 10 should be used for the calculation resolution.

**Example of rounding:**

- Computational resolution: 1000 incr./mm
- Programmed path: 97.3786 mm
- Effective value: 97.379 mm

**Example of programming in the $1/10$ μm range:**

All the linear axes of a machine are to be programmed and traversed within the range of values 0.1 to 1000 μm.

⇒ In order to position accurately to 0.1 μm, the computational resolution must be set to $\geq 10^4$ incr./mm.

⇒ MD10200 $MN_INT_INCR_PER_MM = 10000$ [incr./mm]:
Example of related part program:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N20 G0 X 1.0000 Y 1.0000</td>
<td>; Axes travel to the position X=1.0000 mm, Y=1.0000 mm;</td>
</tr>
<tr>
<td>N25 G0 X 5.0002 Y 2.0003</td>
<td>; Axes travel to the position X=5.0002 mm, Y=2.0003 mm</td>
</tr>
</tbody>
</table>

### 7.2.5 Scaling of physical quantities of machine and setting data

#### Input/output units

Machine and setting data with a physical unit are interpreted in the input/output units stated below depending on whether the metric or inch system is selected:

<table>
<thead>
<tr>
<th>Physical unit</th>
<th>Input/output units for standard basic system</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Metric</td>
</tr>
<tr>
<td>Linear position</td>
<td>1 mm</td>
</tr>
<tr>
<td>Angular position</td>
<td>1 degree</td>
</tr>
<tr>
<td>Linear velocity</td>
<td>1 mm/min</td>
</tr>
<tr>
<td>Angular velocity</td>
<td>1 rpm</td>
</tr>
<tr>
<td>Linear acceleration</td>
<td>1 m/s²</td>
</tr>
<tr>
<td>Angular acceleration</td>
<td>1 rev./s²</td>
</tr>
<tr>
<td>Linear jerk</td>
<td>1 m/s³</td>
</tr>
<tr>
<td>Angular jerk</td>
<td>1 rev./s³</td>
</tr>
<tr>
<td>Time</td>
<td>1 s</td>
</tr>
<tr>
<td>Position controller servo gain</td>
<td>1/s</td>
</tr>
<tr>
<td>Revolitional feedrate</td>
<td>1 mm/rev</td>
</tr>
<tr>
<td>Compensation value linear position</td>
<td>1 mm</td>
</tr>
<tr>
<td>Compensation value angular position</td>
<td>1 degree</td>
</tr>
</tbody>
</table>

Internally in the control, the following units are used, whatever basic system is selected:

<table>
<thead>
<tr>
<th>Physical unit</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear position</td>
<td>1 mm</td>
</tr>
<tr>
<td>Angular position</td>
<td>1 degree</td>
</tr>
<tr>
<td>Linear velocity</td>
<td>1 mm/s</td>
</tr>
<tr>
<td>Angular velocity</td>
<td>1 deg./s</td>
</tr>
<tr>
<td>Linear acceleration</td>
<td>1 mm/s²</td>
</tr>
<tr>
<td>Angular acceleration</td>
<td>1 degree/s²</td>
</tr>
<tr>
<td>Linear jerk</td>
<td>1 mm/s³</td>
</tr>
<tr>
<td>Angular jerk</td>
<td>1 degree/s³</td>
</tr>
<tr>
<td>Time</td>
<td>1 s</td>
</tr>
<tr>
<td>Position controller servo gain</td>
<td>1/s</td>
</tr>
</tbody>
</table>
The user can define different input/output units for machine and setting data. This also requires an adjustment between the newly selected input/output units and the internal units via the following machine data:

- MD10220 $MN_SCALING_USER_DEF_MASK
- MD10230 $MN_SCALING_FACTORS_USER_DEF[n]

The following applies:

Selected I/O unit = (MD10230 $MN_SCALING_FACTORS_USER_DEF[n]) * internal unit

The selected I/O unit, expressed in the internal units 1 mm, 1 degree, and 1 s must therefore be entered in machine data MD10230 $MN_SCALING_FACTORS_USER_DEF[n].

Example 1:

Machine data input/output of the linear velocities is to be in m/min instead of mm/min (initial state).
The scaling factor for the linear velocities is to differ from the standard setting. For this, in machine data:

\[
\text{MD10220 } \$\text{MN SCALING USER DEF MASK}
\]

bit number 2 must be set.

\[
\Rightarrow \text{MD10220 } \$\text{MN SCALING USER DEF MASK} = 'H4'; \text{ (bit no. 2 as hex value)}
\]

The scaling factor is calculated using the following formula:

\[
\text{MD10230 SCALING_FACTORS_USER_DEF[n]} = \frac{\text{Input/output unit selected}}{\text{Internal unit}}
\]

\[
\text{MD10230 SCALING_FACTORS_USER_DEF[n]} = \frac{1 \text{ m/s}^2}{1 \text{ mm/s}^2} = \frac{1000 \text{ mm}}{60 \text{ s}} = \frac{1000}{60} = 16.667;
\]

\[
\Rightarrow \text{MD10230 SCALING_FACTORS_USER_DEF[n]} = 16.667
\]

Index \( n \) defines the "linear velocity" in the "Scaling factors of physical quantities" list.

**Example 2:**

In addition to the change in Example 1, the machine data input/output of linear accelerations must be in ft/s\(^2\) instead of m/s\(^2\) (initial state).

(The internal unit is mm/s\(^2\).)

\[
\Rightarrow \text{MD10220 SCALING_USER_DEF_MASK} = \text{H14 }; \text{ (Bit No. 4 and Bit No. 2 from example 1 as hex value)}
\]

\[
\Rightarrow \text{MD10220 SCALING_FACTORS_USER_DEF[4]} = \frac{1 \text{ ft/s}^2}{1 \text{ mm/s}^2} = \frac{12\times25.4 \text{ mm}}{1 \text{ mm/s}^2} = 304.8 ;
\]

\[
\Rightarrow \text{MD10230 SCALING_FACTORS_USER_DEF[4]} = 304.8
\]

Index 4 defines the "linear acceleration" in the "Scaling factors of physical quantities" list.
7.3 Metric/inch measuring system

7.3.1 Conversion of basic system by part program

Programmable switchover of the system of units

The basic system can be switched over within a part program via the G functions G70/G71/G700/G710 (G group 13). The programmed system of units (G70/G71/G700/G710) and the basic system may be identical or different at any time. Switchover of the system of units within a particular section of the part program could, for example, be used to machine an inch thread on a workpiece within a metric basic system.

The following section of the part program is executed in "metric" in the basic system:

MD10240 $MN_SCALING_SYSTEM_IS_METRIC = 1

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 G71</td>
<td>; Switchover to metric machining</td>
</tr>
<tr>
<td></td>
<td>; The conversion factor does not come into effect because the programmed</td>
</tr>
<tr>
<td></td>
<td>; system of units does not differ from the basic system</td>
</tr>
<tr>
<td></td>
<td>; Metric machining</td>
</tr>
<tr>
<td>....</td>
<td></td>
</tr>
<tr>
<td>N200 G70</td>
<td>; Switchover to machining in inches</td>
</tr>
<tr>
<td></td>
<td>; The conversion factor comes into effect</td>
</tr>
<tr>
<td></td>
<td>; Machining in inches</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N300</td>
<td>; Switchover to metric machining</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Initial state of the G functions

The initial state for the G functions is set for each channel in the following machine data:

MD20150 $MC_GCODE_RESET_VALUES[12] (reset position for G group 13)

When the system of units is changed via the HMI user interface, the reset position is automatically preset for compatibility with the new system of units with G700 or G710.

Displaying length-related data on the HMI

Length-related data are displayed in the parameterized basic system on the HMI:

MD10240 $MN_SCALING_SYSTEM_IS_METRIC (basic system metric)

The following length-related data is displayed in the configured basic system:

- Machine data
- Data in the machine coordinate system
- Tool data
- Zero offsets
The following length-related data is displayed in the programmed system of units:

- Data in the workpiece coordinate system

**Reading in part programs from external sources**

If part programs, including data sets (zero offsets, tool offsets, etc.), programmed in a system of units other than the basic system are read in from an external source, the initial setting must first be changed in machine data MD10240.

**NC/PLC interface signals**

In the case of NC/PLC interface signals containing dimension information, e.g. feedrate for path and positioning axes, data exchange is carried out with the PLC in the configured basic system.

**G functions G700/G710**

The G functions G700/G710 extend the functionality of G70/G71 as follows:

1. The feedrate is interpreted in the programmed system of units:
   - G700: length parameters [inch]; feedrates [inch/min]
   - G710: length parameters [mm]; feedrates [mm/min]

   The programmed feedrate is modal and therefore remains active after subsequent G70/G71/G700/G710 commands. If the feedrate is to apply in the new G70/G71/G700/G710 context, it must be re-programmed.

2. System variables and machine data specifying lengths in the part program are read and written in the programmed system of units.

**Differences during the reading and writing of machine data and system variables**

The following differences exist between G70/G71 and G700/G710 in terms of reading and writing machine data and system variables in the part program:

- **G70/G71:** Reading and writing takes place in the configured basic system.
- **G700/G710:** Reading and writing is performed in the configured system of units.

**Example**

The following part program is executed with an initial metric state:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 R1=0 R2=0</td>
<td></td>
</tr>
<tr>
<td>N120 G01 G70 X1 F1000</td>
<td>; Programmed system of units: inch</td>
</tr>
<tr>
<td>N130 $MA_LUBRICATION_DIST[X]=10</td>
<td>; MD=10 [mm] (basic system)</td>
</tr>
<tr>
<td>N150 IF ($AA_IW[X] &gt; $MA_LUBRICATION_DIST[X])</td>
<td>; [mm] &gt; [mm] (basic system)</td>
</tr>
<tr>
<td>N160 R1=1</td>
<td></td>
</tr>
</tbody>
</table>
Synchronized actions

To ensure, in the case of synchronized actions, that the current part program context does not determine the system of units used in the condition and/or action part, the system of units must be defined within the synchronized action (condition and/or action parts). This is the only way of achieving defined, reproducible behavior in the use of length-related data within a synchronized action.

Example 1
The system of units is not specified within the synchronized action. The X axis is therefore traversed in the system of units of the parameterized initial setting:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 R1=0</td>
<td></td>
</tr>
<tr>
<td>N110 G0 X0 Z0</td>
<td></td>
</tr>
<tr>
<td>N120 WAITP(X)</td>
<td></td>
</tr>
<tr>
<td>N130 ID=1 WHENEVER $R1==1 DO POS[X]=10</td>
<td>X = 10 inch or mm, depending on the rest of the part program</td>
</tr>
<tr>
<td>N140 R1=1</td>
<td></td>
</tr>
<tr>
<td>N150 G71 Z10 F10</td>
<td>Z = 10 mm, X = 10 mm</td>
</tr>
<tr>
<td>N160 G70 Z10 F10</td>
<td>Z = 10 inch, X = 10 inch</td>
</tr>
<tr>
<td>N170 G71 Z10 F10</td>
<td>Z = 10 mm, X = 10 mm</td>
</tr>
<tr>
<td>N180 M30</td>
<td></td>
</tr>
</tbody>
</table>

Example 2
The "metric" system of units is explicitly programmed with G71 within the synchronized action. This means the traversing motion of the X axis takes place in the metric measuring system:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 R1=0</td>
<td></td>
</tr>
<tr>
<td>N110 G0 X0 Z0</td>
<td></td>
</tr>
<tr>
<td>N120 WAITP(X)</td>
<td></td>
</tr>
<tr>
<td>N130 ID=1 WHENEVER $R1==1 DO G71 POS[X]=10</td>
<td>X = 10 mm, independent of the rest of the part program</td>
</tr>
<tr>
<td>N140 R1=1</td>
<td></td>
</tr>
</tbody>
</table>
**Reading and writing of data in the case of G70/G71 and G700/G710 in the part program**

<table>
<thead>
<tr>
<th>Data area</th>
<th>G70 / G71</th>
<th>G700 / G710</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display, decimal places (WCS)</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Display, decimal places (MCS)</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>Feedrates</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>Positional data X, Y, Z</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Interpolation parameters I, J, K</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Circle radius (CR)</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Polar radius (RP)</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Thread pitch</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Programmable FRAME</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Settable FRAMES</td>
<td>G</td>
<td>P</td>
</tr>
<tr>
<td>Basic frames</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>External zero offsets</td>
<td>G</td>
<td>P</td>
</tr>
<tr>
<td>Axial preset offset</td>
<td>G</td>
<td>P</td>
</tr>
<tr>
<td>Operating range limit (G25/G26)</td>
<td>G</td>
<td>P</td>
</tr>
<tr>
<td>Protection zones</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Tool offsets</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>Length-related machine data</td>
<td>G</td>
<td>P</td>
</tr>
<tr>
<td>Length-related setting data</td>
<td>G</td>
<td>P</td>
</tr>
<tr>
<td>Length-related system variables</td>
<td>G</td>
<td>P</td>
</tr>
<tr>
<td>GUD</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>LUD</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>PUD</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>R parameters</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>Siemens cycles</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Jog/handwheel increment factor</td>
<td>G</td>
<td>G</td>
</tr>
</tbody>
</table>

P: Writing/reading is performed in the programmed system of units.
G: Writing/reading takes place in the configured basic system.

**Note**

**Read position data in synchronized actions**

If a system of units has not been explicitly programmed in the synchronized action (condition component and/or action component) **length-related position data** in the synchronized action will always be read in the **parameterized basic system**.
NC-specific conversion factor

The default conversion factor in the machine data:

MD10250 $MN_SCALING_VALUE_INCH (conversion factor for switchover to inch system)

is set to 25.4 for converting from the metric to the inch system. By changing the conversion factor, the control system can also be adapted to customer-specific systems of units.

Axis-specific conversion factor

The default conversion factor in the axis-specific machine data:

MD31200 $MA_SCALING_FACTOR_G70_G71 (conversion factor when G70/G71 is active)

is set to 25.4 for converting from the metric to the inch system. By changing the conversion factor, the control system can also be adapted to customer-specific systems of units for each axis.

7.3.2 Manual switchover of the basic system

General

The control system can operate with the metric or inch system. The initial setting of the system of units (basic system) is defined in the following machine data:

MD10240 $MN_SCALING_SYSTEM_IS_METRIC (basic system metric).

Depending on the basic system, all length-related data is interpreted either as metric or inch measurements.

The relevant softkey on the HMI in the "Machine" operating area is used to change the system of units of the controller.

The system of units is only switched over under the following supplementary conditions:

- MD10260 $MN_CONVERT_SCALING_SYSTEM=1
- MD20110 $MC_RESET_MODE_MASK, bit 0 is set in every channel.
- All channels are in the Reset state.
- Axes do not currently traverse with JOG, DRF, or PLC.
- Constant grinding wheel peripheral speed (GWPS) is not active.

Actions such as part program start or mode change are disabled while the system of units is being switched over.

The actual change in the system of units is made by writing all the necessary machine data and subsequently activating them.
The following machine data are automatically switched over consistently for all configured channels:

- MD10240 $MN_SCALING_SYSTEM_IS_METRIC
- MD20150 $MN_GCODE_RESET_VALUES

**Reset position**
When the system of units is switched over, the reset position of the G group 13 is automatically adapted.

- MD20150 $MC_GCODE_RESET_VALUES[12] = <current system of units>

**Note**
The availability of the softkey and, therefore, its functionality, can be configured using the compatibility machine data:

MD10260 $MN_CONVERT_SCALING_SYSTEM

If several NCUs are linked by NCU-link, the switchover has the same effect on all linked NCUs. If the requirements for a switchover are not fulfilled on one of the connected NCUs, no switchover will take place on any of the NCUs. It is assumed that interpolations between several NCUs will take place on the existing NCUs, whereby the interpolations can provide correct results only if the same systems of units are used.

**References:**
Function Manual, Extended Functions; Several Control Panels on Multiple NCUs, Distributed Systems (B3)

**System data**
When the system of units is switched over, from the user's viewpoint, all length-related specifications are converted to the new system of units automatically.

This includes:

- Positions
- Feedrates
- Accelerations
- Jerk
- Tool offsets
- Programmable, settable and external zero offsets and DRF offsets
- Compensation values
- Protection zones
- Machine data
- Jog and handwheel factors

After the changeover, all of the above data is available in physical quantities.
Data, for which no unique physical units are defined, is not converted automatically.

This includes:

- R parameters
- GUDs (Global User Data)
- LUDs (Local User Data)
- PUDs (Program global User Data)
- Analog inputs/outputs
- Data exchange via FC21

In this case, the user is requested to take into account the currently valid system of units:

```
MD10240 $MN_SCALING_SYSTEM_IS_METRIC
```

The current system of units setting can be read using PLC interface signal:

```
DB10 DBX107.7 (inch system)
```

Using the signal:

```
DB10 DBB71 (change counter, system of units inch/metric)
```

the "system of units change counter" can be read out

- The system of units for sag compensation is configured using:
  
  MD32711 $MA_CEC_SCALING_SYSTEM_METRIC

**References:**
Function Manual Extended Functions; Compensations (K3)

- The system of units for positional data of the indexing axis tables and switching points for software cams is configured in machine data element:

  MD10270 $MN_POS_TAB_SCALING_SYSTEM.

**References:**
Function Manual, Extended Functions; Software Cams, Limit Switching Signals (N3) / Indexing Axes (T1)

**User tool data**

The physical units for user-defined tool and tool cutting edge data can be set in the following machine data:

- MD10290 $MN_CC_TDA_PARAM_UNIT
- MD10292 $MN_CC_TOA_PARAM_UNIT

When the system of units is switched over, all length-related tool data is converted to the new system of units.

**Reference point**

The reference point is retained. It is not necessary to repeat referencing.
**Input resolution and computational resolution**

The input and calculation resolution is defined as an internal number of increments per millimeter.

MD10200 $MN_INT_INCR_PER_MM

The precision of entry of linear positions is limited to the calculation resolution. The product of the programmed position value and the calculation resolution is rounded up to the next integer. To make the rounding clear, powers of 10 should be used for the calculation resolution.

**Example**

1 inch = 25.4 mm ⇒ 0.0001 inch = 0.00254 mm = 2.54 μm = 2540 nm

To be able to program and display the last 40 nm, a value of 100,000 must be parameterized for the input and calculation resolution.

Only with this identical setting for both systems of units is it possible to change the system of units without a significant loss of accuracy. Once MD10200 has been set to this value, it will not need to be changed each time the system of units is switched over.

**JOG and handwheel factor**

The machine data:

MD31090 $MA_JOG_INCR_WEIGHT

consists of two values containing axis-specific increment weighting factors for each of the two systems of units.

Depending on the actual setting in machine data:

MD10240 $MN_SCALING_SYSTEM_IS_METRIC

the controller automatically sets the appropriate value.

The user defines the two increment factors, e.g. for the first axis, during the installation and startup phase:

- Metric:
  MD31090 $MA_JOG_INCR_WEIGHT[0;AX1]=0.001 mm

- Inch:
  MD31090 $MA_JOG_INCR_WEIGHT[1;AX1]=0.00254 mm ≙ 0.0001 Inch

In this way, MD31090 does not have to be written on every inch/metric switchover.

Remaining distances are not accumulated during incremental traversing with JOG when the system of units is changed, since all internal positions always refer to mm.

**Data backup**

Individual data sets that are read from the control and contain data affected by the system of units, are assigned an identifier during reading that indicates the current setting for the system of units:

- MD10260 $MN_CONVERT_SCALING_SYSTEM
- MD10240 $MN_SCALING_SYSTEM_IS_METRIC
The identifier records in which system of units the data have been read out. This ensures that no data sets are read into the control with a system of units other than that which is currently set.

Since the language instruction is also evaluated in part programs, these can also be "protected" against operator errors as described above. You can, therefore, prevent part programs containing only metric data, for example, from running on an inch system of units.

Archive and machine data sets are downward-compatible with the following setting:

MD11220 $MN_INI_FILE_MODE = 2

---

Note

The INCH/METRIC operation is only generated if the compatibility machine data:

MD10260 $MN_CONVERT_SCALING_SYSTEM = TRUE

---

Rounding machine data

All length-related machine data is rounded to the nearest 1 pm when writing in the inch system (MD10240 $MN_SCALING_SYSTEM_IS_METRIC=0 and MD10260 $MN_CONVERT_SCALING_SYSTEM=1), to avoid rounding problems.

This corrects the loss of accuracy resulting from conversion to ASCII when reading out a data backup in the inch system of measurement when the data is read back into the system.

---

### 7.3.3 FGROUP and FGREF

#### Programming

It should be possible to program the effective machining feedrate in the usual way as a path feedrate via the F value in processing procedures where the tool, the workpiece or both are moved by a rotary axis (e.g. laser machining of rotating tubes).

In order to achieve this, it is necessary to specify an effective radius (reference radius) for each of the rotary axes involved. You can do this by programming the modal NC address:

FGREF[<rotary axis>] = <reference radius>

The unit of the reference radius depends on the G70/G71/G700/G710 setting.

In order to include the axes in the calculation of the path feedrate, they must all be specified in the FGROUP command.

In order to ensure compatibility with the behavior with no FGREF programming, the evaluation 1 degree = 1 mm is activated on system powerup and RESET.

This corresponds to a reference radius of:

FGREF = 360 mm / (2π) = 57.296 mm

This default is independent of the active basic system (MD10240 $MN_SCALING_SYSTEM_IS_METRIC) and the currently active G70/G71/G700/ G710 setting.
Special features of the feedrate weighting for rotary axes in FGROUP:

**Program code**

```
N100 FGROUP(X,Y,Z,A)
N110 G1 G91 A10 F100
N120 G1 G91 A10 X0.0001 F100
```

The programmed F value in block N110 is evaluated as a rotary axis feedrate in degrees/min, while the feedrate weighting in block N120 is either 100 inch/min or 100 mm/min, depending on the current inch/metric setting.

**NOTICE**

**Different systems of units**

The FREF factor also works if only rotary axes are programmed in the block. The normal F value interpretation as degree/min applies in this case only if the radius reference corresponds to the FREF default:

- For G71/G710: FREF[A]=57.296
- For G70/G700: FREF[A]=57.296/25.4

**Example**

The following example is intended to demonstrate the effect of FGROUP on the path and path feedrate. The variable $AC_TIME contains the time of the block start in seconds. It can only be used in synchronized actions.

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 G0 X0 A0</td>
<td>Feedrate = 100 mm/min or 100 degrees/min</td>
</tr>
<tr>
<td>N110 FGROUP(X,A)</td>
<td></td>
</tr>
<tr>
<td>N120 G1 G710 F100</td>
<td>Feedrate = 100 mm/min, path = 10 mm, R1 = approx. 6 s</td>
</tr>
<tr>
<td>N130 DO $R1=$AC_TIME</td>
<td></td>
</tr>
<tr>
<td>N140 X10</td>
<td>Feedrate = 100 mm/min, path = 14.14 mm, R2 = approx. 8 s</td>
</tr>
<tr>
<td>N150 DO $R2=$AC_TIME</td>
<td></td>
</tr>
<tr>
<td>N160 X10 A10</td>
<td>Feedrate = 100 degrees/min, path = 10 degrees, R3 = approx. 6 s</td>
</tr>
<tr>
<td>N170 DO $R3=$AC_TIME</td>
<td></td>
</tr>
<tr>
<td>N180 A10</td>
<td>Feedrate = 100 mm/min, path = 10 mm, R4 = approx. 6 s</td>
</tr>
<tr>
<td>N190 DO $R4=$AC_TIME</td>
<td></td>
</tr>
<tr>
<td>N200 X0.001 A10</td>
<td>Feedrate = 2540 mm/min or 100 degrees/min</td>
</tr>
<tr>
<td>N210 G700 F100</td>
<td>Feedrate = 2540 mm/min, path = 254 mm, R5 = approx. 6 s</td>
</tr>
<tr>
<td>N220 DO $R5=$AC_TIME</td>
<td></td>
</tr>
<tr>
<td>N230 X10</td>
<td>Feedrate = 2540 mm/min, path = 254.2 mm, R6 = approx. 6 s</td>
</tr>
<tr>
<td>N240 DO $R6=$AC_TIME</td>
<td></td>
</tr>
<tr>
<td>N250 X10 A10</td>
<td>Feedrate = 2540 mm/min, path = 254.2 mm, R6 = approx. 6 s</td>
</tr>
</tbody>
</table>
### Program code

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N260 DO $R7=$AC_TIME</td>
<td></td>
</tr>
<tr>
<td>N270 A10</td>
<td>; Feedrate = 100 degrees/min, path = 10 degrees, R7 - approx. 6 s</td>
</tr>
<tr>
<td>N280 DO $R8=$AC_TIME</td>
<td></td>
</tr>
<tr>
<td>N290 X0.001 A10</td>
<td>; Feedrate = 2540 mm/min, path = 10 mm, R8 = approx. 0.288 s</td>
</tr>
<tr>
<td>N300 FGREF[A]=360/(2*$PI)</td>
<td>; Set 1 degree = 1 inch via the effective radius</td>
</tr>
<tr>
<td>N310 DO $R9=$AC_TIME</td>
<td></td>
</tr>
<tr>
<td>N320 X0.001 A10</td>
<td>; Feedrate = 2540 mm/min, path = 254 mm, R9 = approx. 6 s</td>
</tr>
<tr>
<td>N330 M30</td>
<td></td>
</tr>
</tbody>
</table>

### Diagnostics

#### Read reference radius

The value of the reference radius of a rotary axis can be read using system variables:

- For the display in the user interface, in synchronized actions or with a preprocessing stop in the part program via the system variable:

  \[ $AA\_FGREF[<axis>] \]  
  Current main run value

- Without preprocessing stop in the part program via the system variable:

  \[ $PA\_FGREF[<axis>] \]  
  Programmed value

If no values are programmed, the default 360 mm / (2π) = 57.296 mm (corresponding to 1 mm per degree) will be read in both variables.

For linear axes, the value in both variables is always 1 mm.

#### Read path axes affecting velocity

The axes involved in path interpolation can be read using system variables:

- For the display in the user interface, in synchronized actions or with a preprocessing stop in the part program via the system variables:

  \[ $AA\_FGROUP[<axis>] \]  
  Returns the value "1" if the specified axis affects the path velocity in the current main run record by means of the basic setting or through \texttt{FGROUP} programming. Otherwise, the variable returns the value "0".

  \[ $AC\_FGROUP\_MASK \]  
  Returns a bit key of the channel axes programmed with \texttt{FGROUP} which are to affect the path velocity.

- Without preprocessing stop in the part program via system variables:
$PA\_FGROUP[<axis>]

Returns the value “1” if the specified axis affects the path velocity by means of the basic setting or through $FGROUP$ programming. Otherwise, the variable returns the value “0”.

$P\_FGROUP\_MASK

Returns a bit key of the channel axes programmed with $FGROUP$ which are to affect the path velocity.

### 7.4 Setpoint/actual-value system

#### 7.4.1 General

**Control loop**

A control loop with the following structure can be configured for every closed-loop controlled axis/spindle:

![Block diagram of a control loop](image)

**Setpoint output**

A setpoint telegram can be output for each axis/spindle. The setpoint output to the actuator is realized from the SINUMERIK 840D sl.
Actual-value acquisition

A maximum of two measuring systems can be connected for each axis/spindle, e.g. a direct measuring system for machining processes with high accuracy requirements and an indirect measuring system for high-speed positioning tasks.

The number of encoders used is recorded in the machine data:

```
MD30200 $MA_NUM_ENCS (number of encoders)
```

In the case of two actual-value branches, the actual value is acquired for both branches.

The active measuring system is always used for position control, absolute value calculation and display. If both measuring systems are activated at the same time by the PLC interface, positioning measuring system 1 is chosen internally by the controller.

Reference point approach is executed by the selected measuring system.

Each position measuring system must be referenced separately.

For an explanation of encoder monitoring, see Section "A3: Axis Monitoring, Protection Zones (Page 79)".

For an explanation of actual-value acquisition compensation functions, see:

References:
Function Manual, Extended Functions; Compensations (K3)

Switching between measuring systems

One can switch between the two measuring systems through the following NC/PLC interface signals:

```
DB31, ... DBX1.5 (position measuring system 1)
DB31, ... DBX1.6 (position measuring system 2)
```

For further information, see Section "A2: Various NC/PLC interface signals and functions (Page 35)".

It is possible to switch over measuring systems at any time, the axes do not have to be stationary to do this. Switchover only takes place if a permissible deviation between the actual values and the two measuring systems has not been violated.

The associated tolerance is entered in the machine data:

```
MD36500 $MA_ENC_CHANGE_TOL (max. tolerance on position actual value switchover)
```

On switchover, the current difference between position measuring system 1 and 2 is traversed immediately.

Monitoring

The permissible deviation between the actual values of the two measuring systems is to be entered in the machine data:

```
MD36510 $MA_ENC_DIFF_TOL
```

For the cyclic comparison of the two measuring systems used, this difference must not be exceeded, as otherwise Alarm 25105 "Measuring systems deviate" is generated.
If the axis is not referenced (at least in the current control measuring system), then the related monitoring is not active if MD36510 = 0 or if neither of the two measuring systems in the axis is active/available.

**Types of actual-value acquisition**

The used encoder type must be defined through the following machine data:

MD30240 $MA_ENC_TYPE (type of actual-value acquisition (actual position value))

**Simulation axes**

The speed control loop of an axis can be simulated for test purposes. The axis "traverses" with a following error, similar to a real axis.

A simulation axis is defined by setting the two following machine data to "0":

MD30130 $MA_CTRLOUT_TYPE[n] (output value of setpoint)

MD30240 $MA_ENC_TYPE[n] (type of actual-value acquisition)

As soon as the standard machine data has been loaded, the axes become simulation axes. The setpoint and actual value can be set to the reference point value with reference point approach.

The machine data:

MD30350 $MA_SIMU_AX_VDI_OUTPUT (output of axis signals with simulation axes)

can be used to define whether the axis-specific interface signals are to be output on the PLC during the simulation.

**Actual-value correction**

If actual-value corrections performed by the NC on the encoder selected for position control do not influence the actual value of another encoder defined in the same axis, then this encoder is to be declared as "independent" via the following machine data:

MD30242 $MA_ENC_IS_INDEPENDENT

Actual-value corrections include the following:

- Modulo treatment
- Reference point approach
- Measuring system comparison
- PRESET
7.4.2 Setpoint and encoder assignment

Setpoint marshalling

The following machine data is relevant for the setpoint assignment of a machine axis.

<table>
<thead>
<tr>
<th>Machine Data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD30100 $MA_CTRLOUT_SEGMENT_NR</td>
<td>PROFIBUS DP / PROFINET (default)</td>
</tr>
</tbody>
</table>

Setpoint assignment: Bus segment

<table>
<thead>
<tr>
<th>System</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>840D sl</td>
<td>5</td>
<td>PROFIBUS DP / PROFINET</td>
</tr>
</tbody>
</table>

MD30110 $MA_CTRLOUT_MODULE_NR

Setpoint assignment: Drive number / module number

<table>
<thead>
<tr>
<th>System</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>840D sl</td>
<td>x</td>
<td>PROFIBUS DP / PROFINET</td>
</tr>
</tbody>
</table>

The logical I/O address of the drive is assigned from MD13050 $MN_DRIVE_LOGIC_ADDRESS[ n ] via the drive number.

The drive number (x) results from the index (n) of MD13050:

\[ x = n + 1 \]

Note

The machine data is of no significance if the drive is simulated (MD30130 $MA_CTRLOUT_TYPE = 0).

MD30120 $MA_CTRLOUT_NR

Setpoint assignment: Setpoint output on drive module/module

<table>
<thead>
<tr>
<th>System</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>840D sl</td>
<td>1</td>
<td>PROFIBUS / PROFINET with PROFIdrive profile</td>
</tr>
</tbody>
</table>

MD30130 $MA_CTRLOUT_TYPE

Setpoint output type

<table>
<thead>
<tr>
<th>System</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>840D sl</td>
<td>0</td>
<td>Simulation (operation without drive)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Setpoint output active</td>
</tr>
</tbody>
</table>

Encoder assignment

The following machine data is relevant for assigning the encoder information of the drive - transferred in the PROFIdrive telegram - to the encoder inputs of the machine axis:

<table>
<thead>
<tr>
<th>Machine Data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD30210 $MA_ENC_SEGMENT_NR[ n ]</td>
<td>PROFIBUS DP / PROFINET</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>840D sl</td>
<td>5</td>
<td>PROFIBUS DP / PROFINET</td>
</tr>
</tbody>
</table>
Actual value assignment: Drive number/measuring circuit number

<table>
<thead>
<tr>
<th>System</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>840D sl</td>
<td>x</td>
<td>The logical I/O address of the drive is assigned from MD13050 $MN_DRIVE_LOGIC_ADDRESS[ n ] via the drive number. The drive number (x) results from the index (n) of MD13050: [ x = n + 1 ]</td>
</tr>
</tbody>
</table>

Actual value assignment: Input on drive module/measuring circuit module

<table>
<thead>
<tr>
<th>System</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>840D sl</td>
<td>x</td>
<td>Number of the encoder interface within the PROFIdrive telegram Examples PROFIdrive telegram 103 [ x = 1 \rightarrow \text{1st encoder interface (G1}_ZSW, \text{G1}_XIST1, \text{G1}_XIST2) ] [ x = 2 \rightarrow \text{2nd encoder interface (G2}_ZSW, \text{G2}_XIST1, \text{G2}_XIST2) ] PROFIdrive telegram 118 [ x = 1 \rightarrow \text{1st encoder interface (G2}_ZSW, \text{G2}_XIST1, \text{G2}_XIST2) ] [ x = 2 \rightarrow \text{2nd encoder interface (G3}<em>ZSW, \text{G3}<em>XIST1, \text{G3}<em>XIST2) ] Note: For SINAMICS S120: - Encoder 1 (G1</em>...): Motor encoder - Encoder 2 (G2</em>...): Direct measuring system - Encoder 3 (G3</em>...): Additional measuring system</td>
</tr>
</tbody>
</table>

Encoder type of the actual value sensing (actual position value)

<table>
<thead>
<tr>
<th>System</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>840D sl</td>
<td>0</td>
<td>Simulation (operation without encoder)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Incremental encoder</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Absolute encoder</td>
</tr>
</tbody>
</table>

Note: Corresponds to PROFIdrive parameter p979
Encoder is independent

<table>
<thead>
<tr>
<th>System</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>840D sl</td>
<td>0</td>
<td>The encoder is not independent.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>The encoder is independent.</td>
</tr>
</tbody>
</table>

If the actual-value corrections, which are made for the encoder selected for the position control, are not to influence the actual value of the second encoder defined in the same axis, then this should be declared as independent.

Actual value corrections are:

- Modulo handling
- Reference point approach
- Measuring system alignment
- PRESET

Example: One axis, two encoders, the 2nd encoder is independent

MD30200 $MA_NUM_ENCS[ AX1 ] = 2
MD30242 $MA_ENC_IS_INDEPENDENT[ 0, AX1 ] = 0
MD30242 $MA_ENC_IS_INDEPENDENT[ 1, AX1 ] = 1

Selection, position measuring system 1 / 2: DB31.DBX1.5 / 1.6

If encoder 1 is selected for closed-loop position control, then the actual value corrections are only performed on this encoder, as encoder 2 is independent.

If encoder 2 is selected for position control, then the actual value corrections are performed on both encoders, as encoder 1 is not independent.

This means that the machine data only has an effect on the passive encoder of a machine axis.

2 The passive encoder is dependent.

The actual encoder value is changed by the active encoder. In combination with MD35102 $MA_REFP_SYNC_ENCS = 1, for reference point approach, the passive encoder is aligned to the active encoder - but is NOT referenced.

In the referencing mode MD34200 $MA_ENC_REFP_MODE = 3 (distance-coded reference marks) the passive encoder is automatically referenced with the next traversing motion after passing the zero mark distance. This is done independent of the actual operating mode setting.

3 The encoder is independent.

For modulo rotary axes, modulo actual value corrections are also performed in the passive encoder.
Note

Machine data index \([ n ]\)

The machine data index \([ n ]\) for encoder assignment has the following meaning:

- \( n = 0 \): First encoder assigned to the machine axis
- \( n = 1 \): Second encoder assigned to the machine axis

The assignment is made using machine data:

- \( \text{MD30220} \ $\text{MA\_ENC\_MODULE\_NR}[ n ] \)
- \( \text{MD30230} \ $\text{MA\_ENC\_INPUT\_NR}[ n ] \)

References

Commissioning Manual, CNC Commissioning: NC, PLC, Drive;
Section: "Communication between the NC and the drive" > "Drives: Assign axis"

7.4.3 Adapting the motor/load ratios

Gear types

The following gear types are available for adapting the mechanical ratios:

<table>
<thead>
<tr>
<th>Gear type</th>
<th>Activation</th>
<th>Adaptation</th>
<th>Installation location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor/load gear</td>
<td>Parameter set</td>
<td>Fixed configuration</td>
<td>Gear unit</td>
</tr>
<tr>
<td>Measuring gear encoder</td>
<td>Power ON</td>
<td>Sensor-dependent</td>
<td>Sensor-side</td>
</tr>
<tr>
<td>Load intermediate gear unit</td>
<td>NewConfig</td>
<td>Load-dependent</td>
<td>Tool-side</td>
</tr>
</tbody>
</table>
Local position of gear unit/encoder

Motor/load gear

The motor/load gear supported by SINUMERIK is configured via the following machine data:
MD31060 $MA\_DRIVE\_AX\_RATIO\_NUMERA$ (numerator load gearbox)
MD31050 $MA\_DRIVE\_AX\_RATIO\_DENOM$ (denominator load gearbox)

The transmission ratio is obtained from the numerator/denominator ratio of both machine data. The associated parameter sets are used automatically as default by the controller to synchronize the position controller with the relevant transmission ratios.

Since a gear stage change is not always carried out automatically, and there are also several ways to request a gear stage change, the position controller is not always incorporated via parameter sets.

Note
For further information about the parameter sets for gear stage change, see Section "S1: Spindles (Page 1199)".

Intermediate gear

Additional, configurable load intermediate gears are also supported by the controller:
MD31066 $MA\_DRIVE\_AX\_RATIO2\_NUMERA$ (intermediate gear numerator)
MD31064 $MA\_DRIVE\_AX\_RATIO2\_DENOM$ (intermediate gear denominator)
Power tools generally have their "own" intermediate gear. Such variable mechanics can be configured by multiplying the active intermediate gearbox and the motor/load gearbox.

<table>
<thead>
<tr>
<th>CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Different gearbox transmission ratios for switching</td>
</tr>
</tbody>
</table>

Unlike the motor/load gear, there is no parameter set for the intermediate gear and, therefore, no way of controlling the time-synchronized switchover to the part program or PLC (NC/PLC interface). Part programming during gear change is, therefore, ruled out. It remains the task of the user to match the synchronization of the relevant changed machine data to the corresponding mechanical switchover and activate it. On switchover during a motion, compensations cannot be ruled out due to jumps in the scaling factors. These are not monitored for violation of the maximum acceleration.

Encoder directly at the tool

Another connection option is possible for a "tool-side encoder" on the intermediate gear, by configuring machine data:

MD31044 $MA_Enc_IS_DIRECT2

Encoder not directly at the tool

The following supplementary conditions apply to a gear change of the intermediate gear in position-control mode:

- The gear ratio to be changed is incorporated in a re-scaling of the encoder information in this case.

In this case, the following applies to axes/spindles in positioning mode:

- A non-abrupt gear change is only possible at zero speed.

To do this, the tool-side position before and after a gear change are set equal for a change in the ratio, since the mechanical position does not (or hardly) change during a gear stage change.

**Recommendation:**

To avoid 21612 "Controller enable reset during motion", changeover should be carried out "only at zero speed". It is still permissible and expedient to switch the axis or spindle to speed-control or follow-up mode before or during a gear change.

Boundary conditions

If the encoder to be used for position control is connected directly at the tool, the gear stage change only affects the physical quantities at the speed interface between the NC and the drive of the motor/load gear. The internal parameter sets are not changed.
Reference point and position reference

<table>
<thead>
<tr>
<th>CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of the position reference</td>
</tr>
</tbody>
</table>

The controller cannot detect all possible situations that can lead to loss of the machine position reference. Therefore, it is the responsibility of the commissioning engineer or user to initiate explicit referencing of zero marker synchronization in such cases.

In the case of gear changes, it is not possible to make a statement about the effect of the reference point or machine position reference on the encoder scaling. In such cases, the controller partially cancels the status "Axis referenced/synchronized".

If the position reference to the machine, tool, etc., has been lost, it must first be restored through appropriate adjustment or referencing of the lost reference point. This is especially important for the functions Travel to fixed stop, Referencing to Bero, Cam and Zero marker.

**Note**

In order to permit new referencing without an interrupting RESET, machine data:

- MD34080 $MA_REFP_MOVE_DIST
- MD34090 $MA_REFP_MOVE_DIST_CORR

are changed over to NewConfig effectiveness.

For further explanations, see Section "R1: Referencing (Page 1149)".

7.4.4 Speed setpoint output

Control direction and travel direction of the feed axes

You must determine the travel direction of the feed axis before starting work.

**Control direction**

Before the position control is started up, the speed controller and current controller of the drive must be started up and optimized.

**Travel direction**

With the machine data:

- MD32100 $MA_AX_MOTION_DIR (travel direction),
- the direction of motion of the axis can be reversed,
- without affecting the control direction of the position control.
**Speed setpoint adjustment**

**SINUMERIK 840D sl**

In the case of speed setpoint comparison, the NC is informed which speed setpoint corresponds to which motor speed in the drive, for parameterizing the axial control and monitoring. This comparison is carried out automatically.

For PROFIBUS DP drives, alternatively, the manual speed setpoint comparison is also possible.

- **Manual comparison**
  
  In the machine data:
  
  MD32250 $MA_RATED_OUTVAL
  
  a value not equal to zero is entered.

**Note**

**Velocity adjustment and maximum speed setpoint**

Owing to the automatic speed setpoint comparison a velocity adjustment is not necessary for SINUMERIK 840D sl!

**Maximum speed setpoint**

For SINUMERIK 840D sl, the maximum speed setpoint is defined as a percentage. 100% means maximum speed setpoint or maximum speed for PROFldeve drives (manufacturer-specific setting parameters in the drive, e.g. p1082 for SINAMICS).

The output of the spindle speed is implemented in the NC for SINUMERIK 840D sl.

Data for five gear stages are realized in the controller.

These stages are defined by a minimum and maximum speed for the stage itself and by a minimum and maximum speed for the automatic gear stage changeover. A new set gear stage is output only if the new programmed speed cannot be traversed in the current gear stage.

With the machine data:

MD36210 $MA_CTRLOUT_LIMIT[n] (maximum speed setpoint)

the speed setpoint is restricted percentage-wise

Values up to 200% are possible.

When the speed is exceeded, an alarm is generated.
However, due to control processes, the axes should not reach their maximum velocity (MD32000 $MA_MAX_AX_VELO) at 100% of the speed setpoint, but at 80% to 95%.

In case of axes, whose maximum speed is attained at around 80% of the speed setpoint range, the default value (80%) of the machine data:

MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

can be taken over.

### 7.4.5 Machine data of the actual value system

#### Axis-specific machine data

To parameterize the actual value system, the following axis-specific machine data should be set:

<table>
<thead>
<tr>
<th>Encoder and parameter set-independent machine data: $MA_</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD30200 NUM_ENCS</td>
<td>Number of encoders</td>
</tr>
<tr>
<td>MD30300 IS_ROT_AX</td>
<td>Rotary axis / spindle</td>
</tr>
<tr>
<td>MD30310 ROT_IS_MODULO</td>
<td>Modulo conversion for rotary axis / spindle</td>
</tr>
<tr>
<td>MD30320 DISPLAY_IS_MODULO</td>
<td>360-degree modulo display for rotary axis or</td>
</tr>
<tr>
<td></td>
<td>spindle</td>
</tr>
<tr>
<td>MD30330 MODULO_RANGE</td>
<td>Size of the modulo range.</td>
</tr>
<tr>
<td>MD30340 MODULO_RANGE_START</td>
<td>Start position of the modulo range.</td>
</tr>
<tr>
<td>MD31030 $MA_LEADSCREW_PITCH</td>
<td>Leadscrew pitch</td>
</tr>
<tr>
<td>MD31064 $MA_DRIVE_AX_RATIO2_DENOM</td>
<td>Intermediate gear denominator</td>
</tr>
<tr>
<td>MD31066 $MA_DRIVE_AX_RATIO2_NUMERA</td>
<td>Intermediate gear numerator</td>
</tr>
<tr>
<td>MD32000 $MA_MAX_AX_VELO</td>
<td>Maximum axis velocity</td>
</tr>
</tbody>
</table>
### Encoder-dependent machine data: $MA_\_\$

<table>
<thead>
<tr>
<th>Encoder-dependent machine data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD30210 ENC_SEGMENT_NR[ n ]</td>
<td>Actual value assignment: Number of bus segments</td>
</tr>
<tr>
<td>MD30220 ENC_MODULE_NR[ n ]</td>
<td>Actual value assignment: Drive number/measuring circuit number</td>
</tr>
<tr>
<td>MD30230 ENC_INPUT_NR[ n ]</td>
<td>Actual value assignment: Input on drive module/measuring circuit module</td>
</tr>
<tr>
<td>MD30240 ENC_TYPE[ n ]</td>
<td>Encoder type of the actual value sensing (position actual value)</td>
</tr>
<tr>
<td>MD30242 ENC_IS_INDEPENDENT[ n ]</td>
<td>Encoder is independent</td>
</tr>
<tr>
<td>MD30244 ENC_MEAS_TYPE[ n ]</td>
<td>Encoder measuring type</td>
</tr>
<tr>
<td>MD30250 ACT_POS_ABS[ n ]</td>
<td>Internal encoder position</td>
</tr>
<tr>
<td>MD30260 ABS_INC_RATIO[ n ]</td>
<td>Absolute encoder: Ratio between the absolute resolution and the incremental resolution</td>
</tr>
<tr>
<td>MD30270 ENC_ABS_BUFFERING[ n ]</td>
<td>Absolute encoder: Traversing range extension</td>
</tr>
<tr>
<td>MD34090 $MA_REFP_MOVE_DIST_CORR[ n ]</td>
<td>Reference point offset</td>
</tr>
<tr>
<td>MD34320 $MA_ENC_INVERS[ n ]</td>
<td>Length measuring system is in the opposite sense</td>
</tr>
</tbody>
</table>

n: Encoder index, with n = 0, 1, ... (1st encoder, 2nd encoder, etc.)

**Note**

Machine data with NEWCONFIG effectiveness criterion can be activated either in the part program with the command `NEWCONF` or via the user interface by pressing a softkey.

---

### 7.4.6 Actual-value resolution

#### 7.4.6.1 Machine data of the actual value resolution

Depending where the measuring system (encoder) is mounted, the following measuring system types should be taken into account:

- **Load-side encoder**: Direct measuring system (DM)
- **Motor-side encoder**: Indirect measuring system (IM)
Parameterizing the actual value resolution depending on the axis type (linear/rotary axis)

The control system calculates the actual value resolution based on the following machine data.

<table>
<thead>
<tr>
<th>Machine data for calculating the actual value resolution</th>
<th>Linear axis</th>
<th>Linear axis</th>
<th>Rotary axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear scale / direct measuring system</td>
<td>Indirect measuring system</td>
<td>Direct measuring system: Machine/tool</td>
<td>Indirect measuring system: Machine/tool</td>
</tr>
<tr>
<td>MD30300 $MA_IS_ROT_AX</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MD31000 $MA_ENC_IS_LINEAR[ n ]</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MD31010 $MA_ENC_GRID_POINT_DIST[ n ]</td>
<td>Spacing</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MD31025 $MA_ENC_PULSE_MULT[ n ]</td>
<td>Encoder multiplication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MD31030 $MA_LEADCSCREW_PITCH</td>
<td>-</td>
<td>mm/rev.</td>
<td>mm/rev.</td>
</tr>
<tr>
<td>MD31040 $MA_ENC_IS_DIRECT[ n ]</td>
<td>- / 1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MD31044 $MA_ENC_IS_DIRECT2[ n ]</td>
<td>- / 1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MD31050 $MA_DRIVE_AX_RATIO_DENOM[ n ]</td>
<td>-</td>
<td>Load rev.</td>
<td>- / 1</td>
</tr>
<tr>
<td>MD31060 $MA_DRIVE_AX_RATIO_NUMERA[ n ]</td>
<td>-</td>
<td>Motor rev. if inf. gear available</td>
<td>- / 1</td>
</tr>
</tbody>
</table>

- Not relevant
1) For distance-coded measuring systems
2) These machine data are not required for encoder matching (path evaluation). However, they must be entered correctly for the setpoint calculation! Otherwise the required servo gain factor ($K_v$) will not be set. The load revolutions are entered into machine data MD31050 $MA_DRIVE_AX_RATIO_DENOM$ and the motor revolutions in machine data MD31060 $MA_DRIVE_AX_RATIO_NUMERA$.
3) The encoder on the motor side is a built-in encoder and, therefore, does not have a measuring gear unit.
The transmission ratio is always 1:1.
G2: Velocities, setpoint / actual value systems, closed-loop control
7.4 Setpoint/actual-value system

Machine data of the actual value resolution
The actual-value resolution results from the design of the machine, whether gearboxes are
available and their gear ratio, the leadscrew pitch for linear axes and the resolution of the
encoder being used. The following machine data must be set for this on the control system:
Encoder and parameter set-independent machine data:
$MA_

Meaning

MD30300 IS_ROT_AX

Axis is a rotary axis / spindle

MD31010 ENC_GRID_POINT_DIST

Distance between reference marks of the
linear scale

MD31030 LEADSCREW_PITCH

Leadscrew pitch

MD31064 DRIVE_AX_RATIO2_DENOM

Denominator of attached gearbox

MD31066 DRIVE_AX_RATIO2_NUMERA

Numerator of attached gearbox

Encoder-dependent machine data: $MA_

Meaning

MD31000 ENC_IS_LINEAR[ n ]

Measuring system is a linear scale

MD31020 ENC_RESOL[ n ]

Encoder pulses per revolution

MD31025 ENC_PULSE_MULT[ n ]

Encoder multiplication (high resolution)

MD31040 ENC_IS_DIRECT[ n ]

Direct or indirect measuring system

MD31044 ENC_IS_DIRECT2[ n ]

Encoders installed at the attached gearbox

MD31070 DRIVE_ENC_RATIO_DENOM[ n ]

Measuring gearbox denominator

MD31080 DRIVE_ENC_RATIO_NUMERA[ n ]

Measuring gearbox numerator

n: Encoder index, with n = 0, 1, ... (1st encoder, 2nd encoder, etc.)

Parameter-set-dependent machine data: $MA_

Meaning

MD31050 $MA_DRIVE_AX_RATIO_DENOM[ m ]

Denominator load gearbox

MD31060 $MA_DRIVE_AX_RATIO_NUMERA[ m ]

Numerator load gearbox

m: Parameter set index, with m = 0, 1, ... (1st parameter set, 2nd parameter set, etc.)

Parameterizing the computational resolution
The ratio of control-internal computational resolution to the actual-value resolution is an
indication of how exactly the values calculated by the control system can be implemented on
the machine.
Computational resolution: Linear axes

350

&RPSXWDWLRQDOUHVROXWLRQ

,QWHUQDOLQFUHPHQWV PP

$FWXDOYDOXHUHVROXWLRQ

(QFRGHULQFUHPHQWV PP

Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2


Computational resolution: Rotary axes

\[
\frac{\text{Computational resolution}}{\text{Actual-value resolution}} = \frac{\text{Internal increments/(degrees)}}{\text{Encoder increments/(degrees)}}
\]

Machine data

<table>
<thead>
<tr>
<th>General machine data: $MN__</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD10200 INT_INCR_PER_MM</td>
<td>Computational resolution for linear positions</td>
</tr>
<tr>
<td>MD10210 INT_INCR_PER_DEG</td>
<td>Computational resolution for angular positions</td>
</tr>
</tbody>
</table>

Recommended setting

The above components and settings that are responsible for the actual-value resolution, should be selected so that the actual-value resolution is higher than the parameterized computational resolution.

\[
\frac{\text{Computational resolution}}{\text{Actual-value resolution}} \leq 1
\]

7.4.6.2 Example: Linear axis with linear scale

The ratio of the internal increments to the encoder increments per mm is calculated as follows:

\[
\frac{\text{Internal increments/mm}}{\text{Encoder increments/mm}} = \frac{\text{ENC_GRID_POINT_DIST}[n] \times \text{INT_INCR_PER_MM}}{\text{ENC_PULSE_MULT}[n]}
\]

Figure 7-4  Linear axis with linear scale
Example: Linear axis with rotary encoder on motor

The ratio of the internal increments to the encoder increments per mm is calculated as follows:

\[
\frac{\text{Internal increments / mm}}{\text{Encoder increments / mm}} = \frac{1}{\text{ENC_RESOL}[n] \times \text{ENC_PULSE_MULTI}[n] \times \text{DRIVE_ENC_RATIO_NUMERA}[n] \times \text{DRIVE_ENC_RATIO_DENOM}[n] \times \text{DRIVE_AX_RATIO_DENOM}[n] \times \text{DRIVE_AX_RATIO_NUMERA}[n] \times \text{LEADSCREW_PITCH} \times \text{INT_INCR_PER_MM}}
\]

Example

Assumptions:
- Rotary encoder on the motor: 2048 pulses/revolution
- Internal pulse multiplication: 2048
- Gearbox, motor / ball screw: 5:1
- Leadscrew pitch: 10 mm/revolution
- Computational resolution: 10000 increments per mm

Machine data

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD30300 $MA_IS_ROT_AX</td>
<td>0</td>
</tr>
<tr>
<td>MD31000 $MA_ENC_IS_LINEAR[0]</td>
<td>0</td>
</tr>
</tbody>
</table>
An encoder increment corresponds to 0.004768 internal increments or 209.731543 encoder increments correspond to an internal increment.

**Example: Linear axis with rotary encoder on the machine**

The ratio of the internal increments to the encoder increments per mm is calculated as follows:

\[
\text{Internal increments/mm} = \frac{1}{2048 \times 2048} \times \frac{1}{1} \times \frac{1}{5} \\
* 10 \text{ mm} \times 10000 \text{ incr./mm} = 0.004768
\]

Figure 7-6  Linear axis with rotary encoder on the machine

The ratio of the internal increments to the encoder increments per mm is calculated as follows:
7.4.6.5 Example: Rotary axis with rotary encoder on motor

Figure 7-7  Rotary axis with rotary encoder on motor

The ratio of the internal increments to the encoder increments per degree is calculated as follows:

\[
\frac{\text{Internal increments / degrees}}{\text{Encoder increments / degrees}} = \frac{360 \text{ degrees}}{\text{ENC_RESOL [n] * ENC_PULSE_MULT[n]}}
\]

- \(\text{DRIVE_ENC_RATIO_NUMERA [n]}\)
- \(\text{DRIVE_ENC_RATIO_DENOM [n]}\)
- \(\text{DRIVE_AX_RATIO_NUMERA [n]}\)
- \(\text{DRIVE_AX_RATIO_DENOM [n]}\)
- \(\text{INT_INCR_PER_DEG}\)
Example

Assumptions:

- Rotary encoder on the motor: 2048 pulses/revolution
- Internal pulse multiplication: 2048
- Gearbox, motor / rotary axis: 5:1
- Computational resolution: 1000 increments per degree

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD30300 $MA_IS_ROT_AX</td>
<td>1</td>
</tr>
<tr>
<td>MD31000 $MA_ENC_IS_LINEAR[0]</td>
<td>0</td>
</tr>
<tr>
<td>MD31040 $MA_ENC_IS_DIRECT[0]</td>
<td>0</td>
</tr>
<tr>
<td>MD31020 $MA_ENC_RESOL[0]</td>
<td>2048</td>
</tr>
<tr>
<td>MD31025 $MA_ENC_PULSE_MULT</td>
<td>2048</td>
</tr>
<tr>
<td>MD31080 $MA_DRIVE_ENC_RATIO_NUMERA[0]</td>
<td>1</td>
</tr>
<tr>
<td>MD31070 $MA_DRIVE_ENC_RATIO_DENOM[0]</td>
<td>1</td>
</tr>
<tr>
<td>MD31060 $MA_DRIVE_AX_RATIO_NUMERA[0]</td>
<td>5</td>
</tr>
<tr>
<td>MD31050 $MA_DRIVE_AX_RATIO_DENOM[0]</td>
<td>1</td>
</tr>
<tr>
<td>MD10210 $MN_INT_INCR_PER_DEG</td>
<td>1000</td>
</tr>
</tbody>
</table>

\[
\Rightarrow \frac{\text{Internal increments/degrees}}{\text{Encoder increments/degrees}} = \frac{360 \text{ degrees}}{2048 \times 2048} \times \frac{1}{1} \times \frac{1}{5}
\]

\[
* 1000 \text{ incr./degrees} = 0.017166
\]

An encoder increment corresponds to 0.017166 internal increments or 58.254689 encoder increments correspond to an internal increment.
7.4.6.6 Example: Rotary axis with rotary encoder on the machine

![Diagram of rotary axis with rotary encoder on the machine](image)

Figure 7-8 Rotary axis with rotary encoder on the machine

The ratio of the internal increments to the encoder increments per degree is calculated as follows:

\[
\frac{\text{Internal increments / degrees}}{\text{Encoder increments / degrees}} = \frac{360 \text{ degrees}}{\text{ENC\_RESOL}[n] \times \text{ENC\_PULSE\_MULT}[n]}
\]

\[
\times \text{DRIVE\_ENC\_RATIO\_NUMER}[n]
\]

\[
\times \text{INT\_INCR\_PER\_DEG}
\]

\[
\text{DRIVE\_ENC\_RATIO\_DENOM}[n]
\]
7.4.6.7 Example: Intermediate gear with encoder on the tool

![Diagram](image)

Figure 7-9 Intermediate gear with encoder directly on the rotating tool

The ratio of the internal increments to the encoder increments per degree is calculated as follows:

\[
\frac{\text{Internal increments / degrees}}{\text{Encoder increments / degrees}} = \frac{360 \text{ degrees}}{\text{ENC_RESOL} [n] \times \text{ENC_PULSE_MULT}[n]}
\]

\[
\times \frac{\text{DRIVE_ENC_RATIO_NUMERA} [n]}{\text{DRIVE_ENC_RATIO_DENOM} [n]}
\]

\[
\times \text{INT_INCR_PER_DEG}
\]

7.5 Closed-loop control

7.5.1 General

Position control of an axis/spindle

The closed-loop control of an axis consists of the current and speed control loop of the drive plus a higher-level position control loop in the NC.

The basic structure of an axis/spindle position control is illustrated below:
For information on the jerk limitation, see Section "B2: Acceleration (Page 243)".

For a description of the feedforward control, backlash, friction compensation, and leadscrew error compensation.

References:
Function Manual Extended Functions; Compensations (K3)

Fine interpolation

Using the fine interpolator (FIPO), the contour precision can be further increased by reducing the staircase effect in the speed setpoint. You can set three different types of fine interpolation:

MD33000 $MA\_FIPO\_TYPE = <FIPO mode>

<table>
<thead>
<tr>
<th>&lt;FIPO mode&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Differential fine interpolation with mean value generation (smoothing) over an IPO cycle</td>
</tr>
<tr>
<td>2</td>
<td>Cubic fine interpolation</td>
</tr>
<tr>
<td>3</td>
<td>Cubic fine interpolation optimized for use with the pre-control for the highest contour precision</td>
</tr>
</tbody>
</table>

Servo gain factor ($K_v$)

In order that few contour deviations occur in the continuous-path mode, a high servo gain factor ($K_v$) is necessary:

MD32200 $MA\_POSCTRL\_GAIN[n]$
However, if the servo gain factor \(K_v\) is too high, instability, overshoot and possibly impermissible high loads on the machine will result.

The maximum permissible servo gain factor \(K_v\) depends on the following:

- Design and dynamics of the drive  
  (rise time, acceleration and braking capacity)
- Machine quality  
  (elasticity, oscillation damping)
- Position control cycle or speed control cycle for active DSC

The servo gain factor \(K_v\) is defined as:

\[
K_v = \frac{\text{Velocity}}{\text{Following error}} \; [\text{m/min}] / [\text{mm}]
\]

Dynamic response adaptation

Axes that interpolate with one another, but with different servo gain factors \(K_v\) can be set to the same following error using the dynamic adaptation function. This allows an optimum contour accuracy to be achieved without loss of control quality by reducing the servo gain factors \(K_v\) to the dynamically weakest axis.

The function is activated via:

MD32900 $MA_DYN_MATCH_ENABLE = 1$ (dynamic response adaptation)

The dynamic response adaptation is realized by entering a new equivalent time constant. It is calculated from the difference in the equivalent time constant of the dynamically weakest axis and the axis to be adapted:

MD32910 $MA_DYN_MATCH_TIME [n] = \text{difference in the equivalent time constant}$
The equivalent time constant of the position control loop is:

<table>
<thead>
<tr>
<th>Axis</th>
<th>Time Constant (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis 1</td>
<td>30</td>
</tr>
<tr>
<td>Axis 2</td>
<td>20</td>
</tr>
<tr>
<td>Axis 3</td>
<td>24</td>
</tr>
</tbody>
</table>

With an equivalent time constant of 30 ms, axis 1 is the dynamically weakest axis.

This results in the following new equivalent time constants for the axes:

- Axis 1: \( \text{MD32910} \ \text{MDA_DYN_MATCH_TIME} = 0 \text{ ms} \)
- Axis 2: \( \text{MD32910} \ \text{MDA_DYN_MATCH_TIME} = 30 \text{ ms} - 20 \text{ ms} = 10 \text{ ms} \)
- Axis 3: \( \text{MD32910} \ \text{MDA_DYN_MATCH_TIME} = 30 \text{ ms} - 24 \text{ ms} = 6 \text{ ms} \)

Approximation formulas for the equivalent time constant of the position control loop of an axis

The equivalent time constant \( T_{\text{equiv}} \) of the position control loop of an axis is approximately calculated depending on the type of feedforward control:

- Without feedforward control:

\[
T_{\text{Spares}} \approx \frac{1}{\text{MD32200 POSCTRL_GAIN} \ [1/\text{s}]} \]
• With speed feedforward control:

\[ T_{S\text{pares}} \approx \text{MD32810 EQUIV\_SPEEDCTRL\_TIME} \]

• For combined torque/speed feedforward control

\[ T_{S\text{pares}} \approx \text{MD32800 EQUIV\_CURRCTRL\_TIME} \]

Note
If dynamic response adaptation is realized for a geometry axis, then all other geometry axes must be set to the same dynamic response.

References:
Commissioning Manual CNC: NCK, PLC, Drives

7.5.2 Parameter sets of the position controller
Six parameter sets per machine axis are available to quickly adapt the position control to the changed machine properties during operation, e.g. a gear change of the spindle, or to adjust the dynamic response to another axis, e.g. during tapping.

Machine data
A parameter set comprises the following machine data:

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier $MA___$</th>
<th>Meaning</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31050</td>
<td>DRIVE_AX_RATIO_DENOM</td>
<td>Denominator load gearbox</td>
<td></td>
</tr>
<tr>
<td>31060</td>
<td>DRIVE_AX_RATIO_NUMERA</td>
<td>Numerator load gearbox</td>
<td></td>
</tr>
<tr>
<td>32200</td>
<td>POSCTRL_GAIN</td>
<td>Servo gain factor ( (K_v) )</td>
<td></td>
</tr>
<tr>
<td>32452</td>
<td>BACKLASH_FACTOR</td>
<td>Backlash compensation</td>
<td></td>
</tr>
<tr>
<td>32610</td>
<td>VELO_FFW_WEIGHT</td>
<td>Feedforward control factor</td>
<td></td>
</tr>
<tr>
<td>36012</td>
<td>STOP_LIMIT_FACTOR</td>
<td>Exact stop coarse/fine factor and zero speed</td>
<td></td>
</tr>
<tr>
<td>32800</td>
<td>EQUIV_CURRCTRL_TIME</td>
<td>Equivalent time constant, current control loop for feedforward control</td>
<td></td>
</tr>
<tr>
<td>32810</td>
<td>EQUIV_SPEEDCTRL_TIME</td>
<td>Equivalent time constant, speed control loop for feedforward control</td>
<td></td>
</tr>
<tr>
<td>32910</td>
<td>DYN_MATCH_TIME</td>
<td>Time constant for dynamic response adaptation</td>
<td></td>
</tr>
<tr>
<td>36200</td>
<td>AX_VELO_LIMIT</td>
<td>Threshold value for velocity monitoring</td>
<td></td>
</tr>
</tbody>
</table>
Tapping, thread cutting

For tapping or thread cutting, the following applies with regard to the parameter sets of axes:

- For machine axes that are not involved in tapping or thread cutting, parameter set 1 (index = 0) is active. The other parameter sets do not have to be taken into account.
- For machine axes that are involved in tapping or thread cutting, the same parameter set number as that of the current gear stage of the spindle is active.

All parameter sets correspond to the gear stages and must therefore be parameterized.

The current parameter set is displayed on the user interface at:

"Operating area switchover" > "Diagnostics" > "Service axis"

Parameter sets during gear stage change

Each gear stage of a spindle is assigned a separate parameter set. The gear stage is selected via the following NC/PLC interface signal:

DB31, ... DBX16.0 - 16.2 = <actual gear stage>

<table>
<thead>
<tr>
<th>Actual gear stage</th>
<th>DB31, ... DBX16.0 - 16.2</th>
<th>Parameter set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Gear stage</td>
<td>000</td>
<td>2</td>
</tr>
<tr>
<td>1. Gear stage</td>
<td>001</td>
<td>2 (Index=1)</td>
</tr>
<tr>
<td>2. Gear stage</td>
<td>010</td>
<td>3</td>
</tr>
<tr>
<td>3. Gear stage</td>
<td>011</td>
<td>4</td>
</tr>
<tr>
<td>4. Gear stage</td>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td>5. Gear stage</td>
<td>101 110 111</td>
<td>6</td>
</tr>
</tbody>
</table>

For further information on gear stages for spindles, see Section "S1: Spindles (Page 1199)".

7.6 Optimization of the control

7.6.1 Position controller, position setpoint filter: Balancing filter

Function

With feedforward control active, the position setpoint is sent through a so-called balancing filter before it reaches the controller itself. It is thus possible to control the speed setpoint to 100% in advance, without resulting in overshoots when positioning.

Activation

The feedforward control variant is selected and so also the filter activated using the axis-specific machine data:
Activating and deactivating via the part program

Part programs can be used to activate and deactivate the feedforward control for all axes, using commands `FFWON` and `FFWOF`.

If the feedforward control of the individual axes should not be influenced by `FFWON/FFWOF`, the setting in the following machine data must be changed for these axes:

\[
\text{MD32630 $MA\_FFW\_ACTIVATION\_MODE$ (activate feedforward control from program)}
\]

Parameterization

**Recommended setting for recommissioning**

If recommissioning, or if default values have been loaded (switch position 1 on commissioning switch and POWER ON), the following machine data default values apply:

- \[\text{MD32620 $MA\_FFW\_MODE = 3$}\]
- \[\text{MD32610 $MA\_VELO\_FFW\_WEIGHT$ (feedforward control factor for the velocity feedforward control) = 1}\]

The balancing time for the speed feedforward control then just has to be adjusted in the following machine data:

\[\text{MD32810 $MA\_EQUIV\_SPEEDCTRL\_TIME$ (equivalent time constant speed control loop for feedforward control)}\]

**Setting the equivalent time constant of the speed control loop (MD32810)**

We recommend that the axis be allowed to move in and out in "AUTOMATIC" mode with a part program and that travel-in to the target position, i.e. the actual position value of the active measuring system, be monitored with the servo trace.

The initial value for setting is the time constant of the speed control loop. This can be read from the reference frequency characteristic of the speed control loop. In the frequent case of a PI controller with speed setpoint smoothing, an approximate equivalent time can be read from drive machine data p1414, p1415, p1416 and p1421.

This start value (e.g. 1.5 ms) is now entered:

\[\text{MD32810 $MA\_EQUIV\_SPEEDCTRL\_TIME = 0.0015}\]

The axis then travels forward and backward and the operator monitors a greatly-magnified characteristic of the actual position value at the target position.
The following rules apply to making manual fine adjustments:

<table>
<thead>
<tr>
<th>Monitoring</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overshoot</td>
<td>Increase MD32810</td>
</tr>
<tr>
<td></td>
<td>Increasing the value slows the axis down but increases the geometric</td>
</tr>
<tr>
<td></td>
<td>contour error on curves; similar to reducing the position controller</td>
</tr>
<tr>
<td></td>
<td>gain: MD32200 $MA_POSCTRL_GAIN. This can be observed on the user</td>
</tr>
<tr>
<td></td>
<td>interface in the operating section &quot;Diagnostics&quot; &gt; &quot;Axis service&quot;</td>
</tr>
<tr>
<td></td>
<td>using the computed $K_V$ value.</td>
</tr>
<tr>
<td>Excessively slow</td>
<td>Reduce MD32810</td>
</tr>
<tr>
<td>approach</td>
<td>Reducing the value speeds the axis up but decreases the geometric</td>
</tr>
<tr>
<td></td>
<td>contour error on curves.</td>
</tr>
</tbody>
</table>

**MD32810 fine adjustment**

Therefore, MD32810 should be assigned as small a value as possible, with the overshoot setting the limit during positioning. Furthermore, the initial value is only modified slightly during fine adjustment, typically by adding or deducting 0.25 ms.

For example, if the initial value is 1.5 ms, the optimum value calculated manually is usually within the range 1.25 ms to 1.75 ms.

In the case of axes equipped with direct measuring systems (load encoders) and strong elasticity, you may possibly accept small overshoots of some micrometers. These can be reduced with the help of the position setpoint filter for dynamic response adaptation (MD32910 $MA_DYN_MATCH_TIME) and for jerk (MD32410 $MA_AX_JERK_TIME), which also reduces the axis speed.

**Identical axis data within an interpolation group**

All the axes within an interpolation group should have identical settings in the following data:

- MD32200 $MA_POSCTRL_GAIN (adapted using MD32910)
- MD32620 $MA_FFW_MODE
- MD32610 $MA_VELO_FFW_WEIGHT
- MD32810 $MA_EQUIV_SPEEDCTRL_TIME (or MD32800 $MA_EQUIV_CURRCTRL_TIME) (dependent on the mechanical system and drive)
- MD32400 $MA_AX_JERK_ENABLE
- MD32402 $MA_AX_JERK_MODE
- MD32410 $MA_AX_JERK_TIME

The servo gain display ($K_V$) of the user interface in operating section "Diagnostics" > "Axis service" is used for checking.

**Non-identical axis data within an interpolation group**

If identical values are not possible for the above data, the following machine data can be used to make an adjustment:

- MD32910 $MA_DYN_MATCH_TIME (time constant of dynamic response adaptation)

This allows the same servo gain value ($K_V$) to be displayed.
Different servo gain display values \((K_v)\) usually point to the following:

- The gear ratios do not match in one or several axes.
- The feedforward control setting data does not match.

**Setting the equivalent time constant of the current control loop (MD32800)**

The activation of the torque feedforward control filter is performed with:

\[
\text{MD32620 } \text{MA\_FFW\_MODE} = 4
\]

The same rules and recommendations apply to setting the time constant of the current control loop \(\text{MD32800 } \text{MA\_EQUIV\_CURRCTRL\_TIME}\) as to the speed feedforward control.

**Limitation to stiff machines**

Experience has shown that this expenditure is only worthwhile in the case of very stiff machines, and requires appropriate experience. The elasticities of the machine are often excited due to the injection of the torque so strongly that the existing vibrations neutralize the gain in contour accuracy.

In this case, it would be worth trying Dynamic Stiffness Control (DSC) as an alternative:

\[
\text{MD32640 } \text{MA\_STIFFNESS\_CONTROL\_ENABLE} = 1
\]

**Note**

If DSC is to be activated \((\text{MD32640} = 1)\), no parameters may be assigned for actual-value inversion in the NC \((\text{MD32110 } \text{MA\_ENC\_FEEDBACK\_POL} = -1)\).

In DSC mode, actual-value inversion may only be undertaken in the drive (SINAMICS parameter p0410).

**Control response with POWER ON, RESET, REPOS, etc.**

In the case of POWER ON and RESET, as well as with "Enable machine data", the setting data of the feedforward control is read in again (see the appropriate values of the machine data).

Mode change, block search and REPOS have no influence on the feedforward control.

**7.6.2 Position controller, position setpoint filter: Jerk filter**

**Function**

In some applications, such as when milling sculptured surfaces, it can be advantageous to smooth the position setpoint curves using the jerk filter to obtain better surfaces by reducing the excitations of machine vibrations. The filter effect of the position setpoints must be as strong as possible without impermissibly affecting contour accuracy. The smoothing behavior of the filter must also be as "symmetrical" as possible, i.e. if the same contour was to be traversed both forward and backward, the characteristic rounded by the filter should be as similar as possible in both directions.
The effect of the filter can be monitored by means of the effective servo gain factor ($K_v$), which is displayed on the "Axis service" screen form. The filtering effect rounds the position setpoints slightly, thus reducing the path accuracy so that with increasing filter time a smaller effective servo gain factor ($K_v$) is displayed.

**Note**

The jerk filter creates a dependent phase offset for each amplitude setting. Only the additional use of the phase filter (see "Position controller, position setpoint filter: Phase filter (Page 368)") permits a transparent setting of the axis dynamic response.

**Activation**

To activate the axial jerk filter setpoint, the following machine data must be set to "TRUE":

MD32400 $MA_{AX}_{JERK\_ENABLE}$ (axial jerk limitation)

**Parameterization**

The axial jerk filter is parameterized via the machine data:

MD32402 $MA_{AX}_{JERK\_MODE}$ (filter type for axial jerk limitation)

MD32410 $MA_{AX}_{JERK\_TIME}$ (time constant for the axial jerk filter)

**Example:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD32400 $MA_{AX}_{JERK_ENABLE}$</td>
<td>1</td>
<td>Enable filter calculation</td>
</tr>
<tr>
<td>MD32402 $MA_{AX}_{JERK_MODE}$</td>
<td>2</td>
<td>Filter mode, moving average value</td>
</tr>
<tr>
<td>MD32410 $MA_{AX}_{JERK_TIME}$</td>
<td>0.02</td>
<td>Set the filter time in seconds (e.g. 20 ms)</td>
</tr>
</tbody>
</table>

If filter mode:MD32402 $MA_{AX}_{JERK\_MODE} = 2$ was not activated previously, "Power On" must be initiated once. Otherwise, "Enable machine data" or "Reset" at the machine control panel are sufficient.

**Note**

Generally, the filter is set using:MD32402 $MA_{AX}_{JERK\_MODE} = 2$. 

G2: Velocities, setpoint / actual value systems, closed-loop control

7.6 Optimization of the control
Fine adjustment
The fine adjustment of the jerk filter is carried out as follows:

1. Assess the traversing response of the axis (e.g. based on positioning processes with servo trace).
2. Modify the filter time in MD32410 $MA_AX_JERK_TIME.
3. Activate the modified time via "Enable machine data" or RESET on the machine control panel.

Deactivation
Disabling the jerk filter:

1. Block the filter calculation:
   MD32400 $MA_AX_JERK_ENABLE = 0
2. Activate the interlock via "Enable machine data" or RESET on the machine control panel.

Boundary conditions
Filter times
The jerk filter is only effective when the time constant (MD32410) is greater than one position control cycle.

Filter effect
- The display of the calculated servo gain factor ($K_v$) in the "Axis service" screen form displays smaller values than would be appropriate based on the filter effect.
- Path accuracy is better than the displayed servo gain ($K_v$) suggests. Therefore, on resetting MD32400 = 1 to MD32400 = 2, the displayed servo gain ($K_v$) can be reduced while retaining the same filter time, although the path accuracy improves.

Axes that are interpolating axes with one another
- Must be set identically.
- Once an optimum value has been identified for these axes, the one with the longest filter time should be used as the setting for all axes within the interpolation group.

References
For further information on jerk limitation at the interpolator level, see Sections "Jerk limitation with single-axis interpolation (SOFTA) (axis-specific) (Page 262)" and "Axis/spindlespecific machine data (Page 291)".
7.6 Optimization of the control

7.6.3 Position controller, position setpoint filter: Phase filter

Function

The axial phase filter setpoint (dead time / delay) implements a pure phase shift with which the setpoint phase response can be influenced. Together with the axial setpoint jerk filter (MD32402_$MA_AX_JERK_MODE[<Axis>] = 2; see section "Position controller, position setpoint filter: Jerk filter (Page 365)"") it makes it possible to adapt the amplitude response and the phase response independently of one another to the dynamically weakest of several axes, which should proceed along a programmed course together.

![Diagram of phase filter setpoint](image)

1. Position control cycle clock: 2 ms
2. Axial phase filter setpoint time constant: 19.2 ms

Figure 7-12 Effect of the axial phase filter setpoint:

Parameter assignment: Time constant

The time constant for the axial phase filter setpoint can be set in the range:

\[(0 \text{ to } 64) \times \text{position control cycle clock (MD10061 $MN\_POSCTRL\_CYCLE\_TIME)}\]

The value of the time constants must be entered in seconds [s] in the machine data:

MD32895 $MA\_DESVAL\_DELAY\_TIME[<Axis>]

Example: Position control cycle clock: 2 ms ⇒ adjustable time constant: 0.0 to 0.128 s

Note

The time constant of the phase filter setpoint delays the axis’ response characteristics for tapping, retractions, and exact stops / block changes, etc. It is therefore recommended to set the time constant as low as possible.

Limiting to a maximum value

A greater time constant than the permitted maximum value (64 \times \text{position control cycle clock}) is limited internally to the maximum value. No message / alarm is displayed.
Parameter assignment: Activation

The function of the axial phase filter setpoint must be activated with the following machine data:

\[ \text{MD32890 $MA\_DESV\_DELAY\_ENABLE[<\text{Axis}>] = "TRUE"} \]

Examples

Assumption: Position control cycle clock = 2 ms

1. \[ \text{MD32890 $MA\_DESV\_DELAY\_ENABLE[<\text{Axis}>] = "FALSE"} \]
   \[ \text{MD32895 $MA\_DESV\_DELAY\_TIME[<\text{Axis}>] = <\text{Time constant}}> \]
   - Phase filter setpoint: Not active
   - Time constant: Irrelevant

2. \[ \text{MD32890 $MA\_DESV\_DELAY\_ENABLE[<\text{Axis}>] = TRUE} \]
   \[ \text{MD32895 $MA\_DESV\_DELAY\_TIME[<\text{Axis}>] = 0.002} \]
   - Phase filter setpoint: Active
   - Time constant: 2 ms ⇒ the setpoint output is delayed by a position control cycle clock.

3. \[ \text{MD32890 $MA\_DESV\_DELAY\_ENABLE[<\text{Axis}>] = TRUE} \]
   \[ \text{MD32895 $MA\_DESV\_DELAY\_TIME[<\text{Axis}>] = 0.256} \]
   - Phase filter setpoint: Active
   - Set time constant: 256 ms;
     maximum possible time constant: 64 * 2 ms = 128 ms
     ⇒ internally effective time constant: 128 ms

Supplementary conditions

**SINUMERIK Safety Integrated**

The phase filter setpoint delays the output of axial setpoints; e.g. for retractions (ESR for Stop E). However, the phase filter setpoint has no influence over shutdown processes; e.g. the SBH activation time.
7.6.4 Position controller: injection of positional deviation

Preconditions

- The function can only be used on axes with two encoders:
  \[ \text{MD30200 $MA\_NUM\_ENCS = 2} \]
  One of the encoders must be parameterized as an indirect measuring system and the other as a direct measuring system:
  - Direct measuring system:
    \[ \text{MD31040 $MA\_ENC\_IS\_DIRECT[1]=1} \]
    The encoder for position actual-value acquisition is connected directly to the machine (load encoder).
  - Indirect measuring system:
    \[ \text{MD31040 $MA\_ENC\_IS\_DIRECT[0]=0} \]
    The encoder for position actual-value acquisition is located on the motor (motor encoder).
- Telegram type 136 or 138 must be configured as standard telegram type for PROFIdrive both in the drive and also in the NC (MD13060 $MN\_DRIVE\_TELEGRAM\_TYPE).

Function

For active injected positional deviation, the difference position between the direct and the indirect measuring system of an axis is determined and in accordance with the weighting-factor setting is applied as additional current setpoint for the feedforward control in the position control cycle. The resulting oscillation damping improves the stability and positioning behavior of the axis.

Application

The function is used for axes with strong tendency to vibrate.

Effectiveness

The function acts only for axes with small natural frequency (to approximately 20 Hz).

Activation/parameterization

The function is activated by specifying the weighting factor:

\[ \text{MD32950 $MA\_POSCTRL\_DAMPING (damping of the speed control loop) = <value>} \]

Value range: -100% ... +100%
An input value “100%” means: A supplementary torque in accordance with SINAMICS parameter p2003 is applied when the determined position difference between the two measuring systems reaches the following value:

- With linear motors: 1 mm
- With linear axis with rotary motor: MD31030 $MA_LEADSCREW_PITCH (leadscrew pitch)
- For rotary axis/spindle: 360 degrees

Standard setting is 0. In this case, the injection of positional deviation is inactive.

Note

The weighting factor MD32950 $MA_POSCTRL_DAMPING can be set on the basis of step responses, for example.

If the control approaches the stability limit (vibration inclination increases), the parameter is too large or the value has the incorrect sign.

### 7.6.5 Position control with proportional-plus-integral-action controller

#### Function

As standard, the core of the position controller is a P controller. It is possible to switch-in an integral component for special applications (such as an electronic gear). The resulting proportional-plus-integral-action controller then corrects the error between setpoint and actual positions down to zero in a finite, settable time period when the appropriate machine data is set accordingly.

<table>
<thead>
<tr>
<th>CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overshootings of the actual position for activated PI controller</strong></td>
</tr>
</tbody>
</table>

In this instance, you must decide whether this effect is admissible for the application in question. Knowledge of the control technology and measurements with servo trace are an absolute prerequisite for using the function. If the appropriate machine data is incorrectly set, then machines could be damaged due to instability.
Procedure

1. First optimize the position control loop as a proportional-action controller using the tools described in the previous subsections.

2. Increase the tolerances of the following machine data while measurements are being taken to determine the quality of the position control with proportional-plus-integral-action controller:
   - MD36020 $MA_POSITIONING_TIME
   - MD36030 $MA_STANDSTILL_POS_TOL
   - MD36040 $MA_STANDSTILL_DELAY_TIME
   - MD36400 $MA_CONTOUR_TOL

3. Activate the position control loop as a proportional-plus-integral-action controller by setting the following machine data:
   MD32220 $MA_POSCTRL_INTEGR_ENABLE ; set value 1
   MD32210 $MA_POSCTRL_INTEGR_TIME ; integral time [sec.]

   Effect of integral time:
   - $T_n \to 0$: The control error is corrected quickly; however, the control loop can become instable.
   - $T_n \to \infty$: Effectiveness of the integral component is almost 0. Behavior of the controller like a pure proportional controller.

4. Find the right compromise for $T_n$ between these two extreme cases for the application.

Note

$T_n$ must not be chosen too near the stability limit because the occurrence of an instability can cause machine damage.

It is therefore recommended to set $T_n$ to no less than 1 s.

Use servo trace to trace the travel-in of an automatic program traveling to and from a target position.

5. Set the servo trace to display the following:
   - Following error
   - Actual velocity
   - Actual position
   - Reference position

6. Reset the tolerance values in the following machine data to the required values, once the optimum value for $T_n$ has been identified:
   - MD36020 $MA_POSITIONING_TIME
   - MD36030 $MA_STANDSTILL_POS_TOL
   - MD36040 $MA_STANDSTILL_DELAY_TIME
   - MD36400 $MA_CONTOUR_TOL
Supplementary conditions

DSC
If the integrator function is used, DSC (Dynamic Stiffness Control) must be switched off.

Example

Setting result after several iterative processes for $K_R$ and $T_n$.

Machine data settings:
- MD32220 $\text{MA\_POSCTRL\_INTEGR\_ENABLE} = 1$
- MD32210 $\text{MA\_POSCTRL\_INTEGR\_TIME} = 0.003$
- MD32200 $\text{MA\_POSCTRL\_GAIN[1]} = 5.0$

Parameter set selection 0

Each of the following quantities - following error, actual velocity, actual position, and position setpoint - has been recorded by servo trace. When traversing in JOG mode, the characteristic of the individual data shown in the following figure was then drawn.
7.7 Data lists

7.7.1 Machine data

7.7.1.1 Displaying machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MM__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9004</td>
<td>DISPLAY_RESOLUTION</td>
<td>Display resolution</td>
</tr>
<tr>
<td>9010</td>
<td>SPIND_DISPLAY_RESOLUTION</td>
<td>Display resolution for spindles</td>
</tr>
<tr>
<td>9011</td>
<td>DISPLAY_RESOLUTION_INCH</td>
<td>Display resolution for INCH system of measurement</td>
</tr>
</tbody>
</table>

7.7.1.2 NC-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>AXCONF_MACHAX_NAME_TAB</td>
<td>Machine axis name</td>
</tr>
<tr>
<td>10050</td>
<td>SYSCLOCK_CYCLE_TIME</td>
<td>System basic cycle</td>
</tr>
<tr>
<td>10070</td>
<td>IPO_SYSCLOCK_TIME_RATIO</td>
<td>Factor for interpolator cycle</td>
</tr>
<tr>
<td>10060</td>
<td>POSCTRL_SYSCLOCK_TIME_RATIO</td>
<td>Factor for position-control cycle</td>
</tr>
<tr>
<td>10200</td>
<td>INT_INCR_PER_MM</td>
<td>Computational resolution for linear positions</td>
</tr>
<tr>
<td>10210</td>
<td>INT_INCR_PER_DEG</td>
<td>Computational resolution for angular positions</td>
</tr>
<tr>
<td>10220</td>
<td>SCALING_USER_DEF_MASK</td>
<td>Activation of scaling factors</td>
</tr>
<tr>
<td>10230</td>
<td>SCALING_FACTORS_USER_DEF</td>
<td>Scaling factors of physical quantities</td>
</tr>
<tr>
<td>10240</td>
<td>SCALING_SYSTEM_IS_METRIC</td>
<td>Basic system metric</td>
</tr>
<tr>
<td>10250</td>
<td>SCALING_VALUE_INCH</td>
<td>Conversion factor for switchover to inch system</td>
</tr>
<tr>
<td>10260</td>
<td>CONVERT_SCALING_SYSTEM</td>
<td>Basic system switchover active</td>
</tr>
<tr>
<td>10270</td>
<td>POS_TAB_SCALING_SYSTEM</td>
<td>Measuring system of position tables</td>
</tr>
<tr>
<td>10290</td>
<td>CC_TDA_PARAM_UNIT</td>
<td>Physical units of the tool data for CC</td>
</tr>
<tr>
<td>10292</td>
<td>CC_TOA_PARAM_UNIT</td>
<td>Physical units of the tool edge data for CC</td>
</tr>
<tr>
<td>13050</td>
<td>DRIVE_LOGIC_ADDRESS</td>
<td>Logical drive addresses</td>
</tr>
<tr>
<td>13060</td>
<td>DRIVE_TELEGRAM_TYPE</td>
<td>Standard message frame type for PROFIBUS DP</td>
</tr>
<tr>
<td>13070</td>
<td>DRIVE_FUNCTION_MASK</td>
<td>DP function used</td>
</tr>
<tr>
<td>13080</td>
<td>DRIVE_TYPE_DP</td>
<td>Drive type PROFIBUS DP</td>
</tr>
</tbody>
</table>

7.7.1.3 Channelspecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20150</td>
<td>GCODE_RESET_VALUES</td>
<td>Initial setting of the G groups</td>
</tr>
</tbody>
</table>
### 7.7.1.4 Axis/spindlespecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>30110</td>
<td>CTRLOUT_MODULE_NR</td>
<td>Setpoint assignment: Drive number</td>
</tr>
<tr>
<td>30120</td>
<td>CTRLOUT_NR</td>
<td>Setpoint assignment: Setpoint output on drive module</td>
</tr>
<tr>
<td>30130</td>
<td>CTRLOUT_TYPE</td>
<td>Output type of setpoint</td>
</tr>
<tr>
<td>30200</td>
<td>NUM_ENCS</td>
<td>Number of encoders</td>
</tr>
<tr>
<td>30220</td>
<td>ENC_MODULE_NR</td>
<td>Actual value assignment: Drive module number</td>
</tr>
<tr>
<td>30230</td>
<td>ENC_INPUT_NR</td>
<td>Actual value assignment: Input on the drive module</td>
</tr>
<tr>
<td>30240</td>
<td>ENC_TYPE</td>
<td>Type of actual-value acquisition (position actual value)</td>
</tr>
<tr>
<td>30242</td>
<td>ENC_IS_INDEPENDENT</td>
<td>Encoder is independent</td>
</tr>
<tr>
<td>30300</td>
<td>IS_ROT_AX</td>
<td>Rotary axis</td>
</tr>
<tr>
<td>31000</td>
<td>ENC_IS_LINEAR</td>
<td>Direct measuring system (linear scale)</td>
</tr>
<tr>
<td>31010</td>
<td>ENC_GRID_POINT_DIST</td>
<td>Distance between reference marks on linear scales</td>
</tr>
<tr>
<td>31020</td>
<td>ENC_RESOL</td>
<td>Encoder pulses per revolution</td>
</tr>
<tr>
<td>31030</td>
<td>LEADSCREW_PITCH</td>
<td>Leadscrew pitch</td>
</tr>
<tr>
<td>31040</td>
<td>ENC_IS_DIRECT</td>
<td>Encoder is connected directly to the machine</td>
</tr>
<tr>
<td>31044</td>
<td>ENC_IS_DIRECT2</td>
<td>Encoder on intermediate gear</td>
</tr>
<tr>
<td>31050</td>
<td>DRIVE_AX_RATIO_DENOM</td>
<td>Denominator load gearbox</td>
</tr>
<tr>
<td>31060</td>
<td>DRIVE_AX_RATIO_NUMERA</td>
<td>Numerator load gearbox</td>
</tr>
<tr>
<td>31064</td>
<td>DRIVE_AX_RATIO2_DENOM</td>
<td>Intermediate gear denominator</td>
</tr>
<tr>
<td>31066</td>
<td>DRIVE_AX_RATIO2_NUMERA</td>
<td>Intermediate gear numerator</td>
</tr>
<tr>
<td>31070</td>
<td>DRIVE_ENC_RATIO_DENOM</td>
<td>Measuring gearbox denominator</td>
</tr>
<tr>
<td>31080</td>
<td>DRIVE_ENC_RATIO_NUMERA</td>
<td>Measuring gearbox numerator</td>
</tr>
<tr>
<td>31090</td>
<td>JOG_INCR_WEIGHT</td>
<td>Weighting of increment for INC/handwheel</td>
</tr>
<tr>
<td>31200</td>
<td>SCALING_FACTOR_G70_G71</td>
<td>Factor for converting values when G70/G71 is active</td>
</tr>
<tr>
<td>32000</td>
<td>MAX_AX_VELO</td>
<td>Maximum axis velocity</td>
</tr>
<tr>
<td>32100</td>
<td>AX_MOTION_DIR</td>
<td>Travel direction</td>
</tr>
<tr>
<td>32110</td>
<td>ENC_FEEDBACK_POL</td>
<td>Sign actual value (feedback polarity)</td>
</tr>
<tr>
<td>32200</td>
<td>POSCTRL_GAIN</td>
<td>Servo gain factor ($K_v$)</td>
</tr>
<tr>
<td>32210</td>
<td>POSCTRL_INTEGR_TIME</td>
<td>Integrator time position controller</td>
</tr>
<tr>
<td>32220</td>
<td>POSCTRL_INTEGR_ENABLE</td>
<td>Activation of integral component of position controller</td>
</tr>
<tr>
<td>32250</td>
<td>RATED_OUTVAL</td>
<td>Rated output voltage</td>
</tr>
<tr>
<td>32260</td>
<td>RATED_VELO</td>
<td>Rated motor speed</td>
</tr>
<tr>
<td>32450</td>
<td>BACKLASH</td>
<td>Backlash</td>
</tr>
<tr>
<td>32500</td>
<td>FRICT_COMP_ENABLE</td>
<td>Friction compensation active</td>
</tr>
<tr>
<td>32610</td>
<td>VELO_FFW_WEIGHT</td>
<td>Feedforward control factor for speed feedforward control</td>
</tr>
<tr>
<td>32620</td>
<td>FFW_MODE</td>
<td>Feedforward control mode</td>
</tr>
<tr>
<td>32630</td>
<td>FFW_ACTIVATION_MODE</td>
<td>Activate feedforward control from program</td>
</tr>
<tr>
<td>32650</td>
<td>AX_INERTIA</td>
<td>Moment of inertia for torque feedforward control</td>
</tr>
<tr>
<td>32652</td>
<td>AX_MASS</td>
<td>Axis mass for torque precontrol</td>
</tr>
<tr>
<td>Number</td>
<td>Identifier: $MA__</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>32711</td>
<td>CEC_SCALING_SYSTEM_METRIC</td>
<td>System of measurement of sag compensation</td>
</tr>
<tr>
<td>32800</td>
<td>EQUIV_CURRCTRL_TIME</td>
<td>Equivalent time constant current control loop for feed-forward control</td>
</tr>
<tr>
<td>32810</td>
<td>EQUIV_SPEEDCTRL_TIME</td>
<td>Equivalent time constant speed control loop for feed-forward control</td>
</tr>
<tr>
<td>32890</td>
<td>DESVAL_DELAY_ENABLE</td>
<td>Axial phase filter setpoint</td>
</tr>
<tr>
<td>32895</td>
<td>DESVAL_DELAY_TIME</td>
<td>Time constant for the axial phase filter setpoint</td>
</tr>
<tr>
<td>32900</td>
<td>DYN_MATCH_ENABLE</td>
<td>Dynamics matching</td>
</tr>
<tr>
<td>32910</td>
<td>DYN_MATCH_TIME[n]</td>
<td>Time constant for dynamic response adaptation</td>
</tr>
<tr>
<td>32930</td>
<td>POSCTRL_OUT_FILTER_ENABLE</td>
<td>Activation of low-pass filter at position controller output</td>
</tr>
<tr>
<td>32950</td>
<td>POSCTRL_DAMPING</td>
<td>Damping of the speed control loop</td>
</tr>
<tr>
<td>33000</td>
<td>FIPO_TYPE</td>
<td>Fine interpolator type</td>
</tr>
<tr>
<td>34320</td>
<td>ENC_INVERS[n]</td>
<td>Length measuring system is inverse</td>
</tr>
<tr>
<td>35100</td>
<td>SPIND_VELO_LIMIT</td>
<td>Maximum spindle speed</td>
</tr>
<tr>
<td>36200</td>
<td>AX_VELO_LIMIT[n]</td>
<td>Threshold value for velocity monitoring</td>
</tr>
<tr>
<td>36210</td>
<td>CTRLOUT_LIMIT[n]</td>
<td>Maximum speed setpoint</td>
</tr>
<tr>
<td>36400</td>
<td>AX_JERK_ENABLE</td>
<td>Axial jerk limitation</td>
</tr>
<tr>
<td>36410</td>
<td>AX_JERK_TIME</td>
<td>Time constant for axial jerk filter</td>
</tr>
<tr>
<td>36500</td>
<td>ENC_CHANGE_TOL</td>
<td>Max. tolerance for position actual-value switchover</td>
</tr>
<tr>
<td>36510</td>
<td>ENC_DIFF_TOL</td>
<td>Measuring system synchronism tolerance</td>
</tr>
<tr>
<td>36700</td>
<td>ENC_COMP_ENABLE[n]</td>
<td>Interpolatory compensation</td>
</tr>
</tbody>
</table>
H2: Auxiliary function outputs to PLC

8.1 Brief description

8.1.1 Function

Auxiliary functions permit activation of the system functions of the NCK and PLC user functions. Auxiliary functions can be programmed in:

- Part programs
- Synchronized actions
- User cycles

For detailed information on the use of auxiliary function outputs in synchronized actions, see:

References:
Function Manual, Synchronized Actions

Predefined auxiliary functions

Predefined auxiliary functions activate system functions. The auxiliary function is also output to the NC/PLC interface.

The following auxiliary functions are predefined:

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Additional</td>
<td>M30</td>
<td>End of program</td>
</tr>
<tr>
<td>S</td>
<td>Spindle</td>
<td>S100</td>
<td>Spindle speed 100 (e.g. rpm)</td>
</tr>
<tr>
<td>T</td>
<td>Tool number</td>
<td>T3</td>
<td>Tool number 3</td>
</tr>
<tr>
<td>D, DL</td>
<td>Tool offset</td>
<td>D1</td>
<td>Tool cutting edge number 1</td>
</tr>
<tr>
<td>F</td>
<td>Feedrate</td>
<td>F1000</td>
<td>Feedrate 1000 (e.g. mm/min)</td>
</tr>
</tbody>
</table>

Userdefined auxiliary functions

User-defined auxiliary functions are either extended predefined auxiliary functions or user-specific auxiliary functions.

Extension of predefined auxiliary functions

Extension of predefined auxiliary functions refers to the "address extensions" parameter. The address extension defines the number of the spindle to which the auxiliary function applies. The spindle function M3 (spindle right) is predefined for the master spindle of a channel. If a
2nd spindle is assigned to a channel, a corresponding user-defined auxiliary function must be defined that extends the predefined auxiliary function.

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Additional function</td>
<td>M2=3</td>
<td>2nd spindle: Spindle right</td>
</tr>
<tr>
<td>S</td>
<td>Spindle function</td>
<td>S2=100</td>
<td>2nd spindle: Spindle speed = 100 (e.g. rpm)</td>
</tr>
<tr>
<td>T</td>
<td>Tool number</td>
<td>T2=3</td>
<td></td>
</tr>
</tbody>
</table>

**User-specific auxiliary functions**

User-specific auxiliary functions do not activate system functions. User-specific auxiliary functions are output to the NC/PLC interface only. The functionality of the auxiliary functions must be implemented by the machine manufacturer / user in the PLC user program.

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>Auxiliary function</td>
<td>H2=5</td>
<td>User-specific function</td>
</tr>
</tbody>
</table>

1) Recommendation

### 8.1.2 Definition of an auxiliary function

An auxiliary function is defined by the following parameters:

- **Type, address extension and value**
  The three parameters are output to the NC/PLC interface.

- **Output behavior**
  The auxiliary function-specific output behavior defines for how long an auxiliary function is output to the NC/PLC interface and when it is output relative to the traversing motion programmed in the same part program block.

- **Group assignment**
  An auxiliary function can be assigned to a particular auxiliary function group. The output behavior can be defined separately for each auxiliary function group. This becomes active if no auxiliary function-specific output behavior has been defined. Group membership also affects output of an auxiliary function after block search.

For more detailed information on auxiliary function output to the NC/PLC interface, see Section "P3: Basic PLC program for SINUMERIK 840D sI (Page 821)".

### 8.1.3 Overview of the auxiliary functions

**M functions**

<table>
<thead>
<tr>
<th>M (special function)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address extension</td>
</tr>
<tr>
<td>Range of values</td>
</tr>
<tr>
<td>0 (implicit)</td>
</tr>
</tbody>
</table>

Basic Functions

Function Manual, 01/2015, 6FC5397-0BP40-5BA2
### M (special function)

**Remarks:**
The address extension is 0 for the range between 0 and 99.
Mandatory without address extension: M0, M1, M2, M17, M30

<table>
<thead>
<tr>
<th>Range of values</th>
<th>Meaning</th>
<th>Range of values</th>
<th>Type</th>
<th>Meaning</th>
<th>Number 8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ... 20</td>
<td>Spindle number</td>
<td>1 ... 99</td>
<td>INT</td>
<td>Function</td>
<td>5</td>
</tr>
</tbody>
</table>

**Remarks:**
M3, M4, M5, M19, M70 with address extension as the spindle number. (e.g. M2=5; spindle stop for spindle 2).
Without an address extension, the function influences the master spindle.

<table>
<thead>
<tr>
<th>Range of values</th>
<th>Meaning</th>
<th>Range of values</th>
<th>Type</th>
<th>Meaning</th>
<th>Number 8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ... 99</td>
<td>Any</td>
<td>100 ... 2147483647</td>
<td>INT</td>
<td>Function</td>
<td>5</td>
</tr>
</tbody>
</table>

**Remarks:**
User-specific M functions.

8) See "Meaning of footnotes" at the end of the overview.

### Use

Controlling machine functions in synchronism with the part program.

### Further information

- The following M functions have a predefined meaning: M0, M1, M2, M17, M30 M3, M4, M5, M6, M19, M70, M40, M41, M42, M43, M44, M45.
- For each M function (M0 - M99), there is a dynamic signal at the NC/PLC interface that indicates the validity (new output) of the M function. In addition, 64 further signals can be assigned for user M functions (see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 821)").
- For subprograms, machine data can be used to set whether an output of the M function should be undertaken for the end of the part program M17, M2 and M30 to the PLC: MD20800 $MC_SPF_END_TO_VDI (subprogram end to PLC)
- For the predefined M function M40 – M45, only limited redefinition of the output specification is possible.
The predefined auxiliary functions M0, M1, M17, M30, M6, M4, M5 cannot be redefined.

M function-specific machine data:
- MD10800 $MN_EXTERN_CHAN_SYNC_M_NO_MIN
- MD10802 $MN_EXTERN_CHAN_SYNC_M_NO_MAX
- MD10804 $MN_EXTERN_M_NO_SET_INT
- MD10806 $MN_EXTERN_M_NO_DISABLE_INT
- MD10814 $MN_EXTERN_M_NO_MAC_CYCLE
- MD10815 $MN_EXTERN_M_NO_MAC_CYCLE_NAME
- MD20094 $MC_SPIND_RIGID_TAPPING_M_NR
- MD20095 $MC_EXTERN_RIGID_TAPPING_M_NR
- MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO
- MD22000 $MC_AUXFU_M_SYNC_TYPE
- MD22530 $MC_TOCARR_CHANGE_M_CODE
- MD22532 $MC_GEOAX_CHANGE_M_CODE
- MD22534 $MC_TRAFO_CHANGE_M_CODE
- MD22560 $MC TOOL CHANGE M_CODE

S functions

<table>
<thead>
<tr>
<th>Address extension</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of values</td>
<td>Meaning</td>
</tr>
<tr>
<td>0 ... 20</td>
<td>Spindle number 5)</td>
</tr>
</tbody>
</table>

Remarks:
The master spindle of the channel is addressed if no address extension is specified.

3), 5, 8) See "Meaning of footnotes" at the end of the overview.

Use
- Spindle speed.

Further information
- S functions are assigned to auxiliary function group 3 by default.
- Without an address extension, the S functions refer to the master spindle of the channel.
- S function-specific machine data:
  - MD22210 $MC_AUXFU_S_SYNC_TYPE (output time of the S functions)

H functions

<table>
<thead>
<tr>
<th>Address extension</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of values</td>
<td>Meaning</td>
</tr>
</tbody>
</table>

380 Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2
H (aux. function)

<table>
<thead>
<tr>
<th>Value</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ... 99</td>
<td>INT</td>
<td>Any</td>
</tr>
<tr>
<td>-2147483648</td>
<td>REAL</td>
<td>Any</td>
</tr>
<tr>
<td>+2147483647</td>
<td>REAL</td>
<td>Any</td>
</tr>
<tr>
<td>0 ... ±3.4028 exp38</td>
<td>REAL</td>
<td>Any</td>
</tr>
</tbody>
</table>

Remarks:
The functionality must be implemented by the user in the PLC user program.

2) 3) 4) 8) See "Meaning of footnotes" at the end of the overview.

Use
User-specific auxiliary functions.

Further information
- H function-specific machine data:
  MD22110 $MC_AUXFU_H_TYPE_INT (type of H-auxiliary function is an integer)
  MD22230 $MC_AUXFU_H_SYNC_TYPE (output time of the H functions)

T functions

<table>
<thead>
<tr>
<th>Value</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ... 12</td>
<td>INT</td>
<td>Spindle number (with active tool management)</td>
</tr>
<tr>
<td>0 ... 32000</td>
<td>INT</td>
<td>Selection of the tool</td>
</tr>
</tbody>
</table>

Remarks:
Tool names are not output to the PLC. 1)

1) 5) 6) 8) See "Meaning of footnotes" at the end of the overview.

Use
Tool selection.

Further information
- Identification of the tools, optionally via tool number or location number (see Section "W1: Tool offset (Page 1363)").

References:
Function Manual Tool Management

- When T0 is selected, the current tool is removed from the toolholder but not replaced by a new tool (default setting).
- T function-specific machine data:
  MD22220 $MC_AUXFU_T_SYNC_TYPE (output time of the T functions)
### D functions

<table>
<thead>
<tr>
<th>Address extension</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of values</td>
<td>Meaning</td>
</tr>
<tr>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

Remarks:
Clearing the tool offset with D0. Default is D1.

Use
Selection of the tool offset.

Further information
- Initial setting: D1
- After a tool change, the default tool cutting edge can be parameterized via: MD20270 $MC_CUTTING_EDGE_DEFAULT (basic position of the tool cutting edge without programming)
- Deselection of the tool offset: D0
- D function-specific machine data: MD22250 $MC_AUXFU_D_SYNC_TYPE (output time of the D functions)

### DL functions

<table>
<thead>
<tr>
<th>Address extension</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of values</td>
<td>Meaning</td>
</tr>
<tr>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

Remarks:
The additive tool offset selected with DL refers to the active D number.

Use
Selection of the additive tool offset with reference to an active tool offset.

Further information
- Initial setting: DL = 0
- DL values cannot be output to the PLC via synchronized actions.
- Default setting of the additive tool offset without an active DL function: MD20272 $MC_SUMCORR_DEFAULT (basic setting of the additive offset without a program)
Deselection of the additive tool offset: DL = 0

DL function-specific machine data:
MD22252 $MC_AUXFU_DL_SYNC_TYPE (output time DL functions)

F functions

<table>
<thead>
<tr>
<th>Address extension</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of values</td>
<td>Meaning</td>
</tr>
<tr>
<td>- - -</td>
<td>- - -</td>
</tr>
</tbody>
</table>

Remarks:

See "Meaning of footnotes" at the end of the overview.

Use
Path velocity.

Further information
- F function-specific machine data:
  MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of F functions)

FA functions

<table>
<thead>
<tr>
<th>Address extension</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of values</td>
<td>Meaning</td>
</tr>
<tr>
<td>1 - 31</td>
<td>Axis number</td>
</tr>
</tbody>
</table>

Remarks:

See "Meaning of footnotes" at the end of the overview.

Use
Axial velocity.

Further information
- F function-specific machine data:
  MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of F functions)
Meaning of footnotes

1) If tool management is active, neither a T change signal nor a T word is output to the interface (channel).

2) The type for the values can be selected by the user via MD22110 $MC_AUXFU_H_TYPE_INT.

3) Because of the limited display options on the operator panel screens, the REAL type values displayed are restricted to:
   –999 999 999.9999 to 999 999 999.9999
   The NC calculates internally but with complete accuracy.

4) The REAL values are rounded and output to the PLC when setting the machine data:
   MD22110 $MC_AUXFU_H_TYPE_INT = 1 (type of H-auxiliary functions is an integer)
   The PLC user program must interpret the value transferred according to the machine data setting.

5) If the tool management is active, the meaning of the address extension can be parameterized. Address extension = 0 means the value must be replaced by that of the master spindle number, i.e. it is equivalent to not programming the address extension.
   Auxiliary functions M19 "Position spindle" collected during a block search are not output to the PLC.

6) M6: Range of values of the address extension:
   - without tool management: 0 ... 99
   - with tool management: 0 ... maximum spindle number
   0: to be replaced by the value of the master spindle number or master toolholder

7) If tool management is active, the auxiliary function M6 "Tool change" can only be programmed once in a part program block, irrespective the address extensions that are programmed.

8) Maximum number of auxiliary functions per part program block.

8.2 Predefined auxiliary functions

Function

Every pre-defined auxiliary function is assigned to a system function and cannot be changed.
If a pre-defined auxiliary function is programmed in a part program/cycle, then this is output to the PLC via the NC/PLC interface and the corresponding system function is executed in the NCK.
Definition of a predefined auxiliary function

The parameters of the predefined auxiliary function are stored in machine data and can be changed in some cases. All machine data, which are assigned to an auxiliary function, have the same index \(<n>\).

- MD22040 $MC_AUXFU_PREDEF_GROUP[<n>] (group assignment of predefined auxiliary functions)
- MD22050 $MC_AUXFU_PREDEF_TYPE[<n>] (type of predefined auxiliary functions)
- MD22060 $MC_AUXFU_PREDEF_EXTENSION[<n>] (address extension for predefined auxiliary functions)
- MD22070 $MC_AUXFU_PREDEF_VALUE[<n>] (value of predefined auxiliary functions)
- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>] (output behavior of predefined auxiliary functions)

8.2.1 Overview: Predefined auxiliary functions

Significance of the parameters listed in the following tables:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index (&lt;n&gt;)</td>
<td>Machine data index of the parameters of an auxiliary function</td>
</tr>
<tr>
<td>Type</td>
<td>MD22050 $MC_AUXFU_PREDEF_TYPE[&lt;n&gt;]</td>
</tr>
<tr>
<td>Address extension</td>
<td>MD22060 $MC_AUXFU_PREDEF_EXTENSION[&lt;n&gt;]</td>
</tr>
<tr>
<td>Value</td>
<td>MD22070 $MC_AUXFU_PREDEF_VALUE[&lt;n&gt;]</td>
</tr>
<tr>
<td>Group</td>
<td>MD22040 $MC_AUXFU_PREDEF_GROUP[&lt;n&gt;]</td>
</tr>
</tbody>
</table>

Predefined auxiliary functions

### General auxiliary functions, Part 1

<table>
<thead>
<tr>
<th>System function</th>
<th>Index (&lt;n&gt;)</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>0</td>
<td>M</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Conditional stop</td>
<td>1</td>
<td>M</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>End of subprogram</td>
<td>2</td>
<td>M</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>M</td>
<td>0</td>
<td>17</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>M</td>
<td>0</td>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>Tool change</td>
<td>5</td>
<td>M</td>
<td>(0)</td>
<td>6  (^1)</td>
<td>(1)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 1

<table>
<thead>
<tr>
<th>System function</th>
<th>Index (&lt;n&gt;)</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>6</td>
<td>M</td>
<td>1</td>
<td>3</td>
<td>(2)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>7</td>
<td>M</td>
<td>1</td>
<td>4</td>
<td>(2)</td>
</tr>
</tbody>
</table>

\(^1\) In this case, \(n\) is \(0\) or \(1\).
## 8.2 Predefined auxiliary functions

### Spindle-specific auxiliary functions, spindle 1

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle stop</td>
<td>8</td>
<td>M</td>
<td>1</td>
<td>5</td>
<td>(2)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>9</td>
<td>M</td>
<td>1</td>
<td>19</td>
<td>(2)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>10</td>
<td>M</td>
<td>1</td>
<td>70 2)</td>
<td>(2)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>11</td>
<td>M</td>
<td>1</td>
<td>40</td>
<td>(4)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>12</td>
<td>M</td>
<td>1</td>
<td>41</td>
<td>(4)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>13</td>
<td>M</td>
<td>1</td>
<td>42</td>
<td>(4)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>14</td>
<td>M</td>
<td>1</td>
<td>43</td>
<td>(4)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>15</td>
<td>M</td>
<td>1</td>
<td>44</td>
<td>(4)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>16</td>
<td>M</td>
<td>1</td>
<td>45</td>
<td>(4)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>17</td>
<td>S</td>
<td>1</td>
<td>-1</td>
<td>(3)</td>
</tr>
</tbody>
</table>

### General auxiliary functions, Part 2

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feedrate</td>
<td>18</td>
<td>F</td>
<td>0</td>
<td>-1</td>
<td>(1)</td>
</tr>
<tr>
<td>Cutting edge selection</td>
<td>19</td>
<td>D</td>
<td>0</td>
<td>-1</td>
<td>(1)</td>
</tr>
<tr>
<td>DL</td>
<td>20</td>
<td>L</td>
<td>0</td>
<td>-1</td>
<td>(1)</td>
</tr>
<tr>
<td>Tool selection</td>
<td>21</td>
<td>T</td>
<td>(0)</td>
<td>-1</td>
<td>(1)</td>
</tr>
<tr>
<td>Stop (associated)</td>
<td>22</td>
<td>M</td>
<td>0</td>
<td>-1 3)</td>
<td></td>
</tr>
<tr>
<td>Conditional stop (associated)</td>
<td>23</td>
<td>M</td>
<td>0</td>
<td>-1 4)</td>
<td>1</td>
</tr>
<tr>
<td>End of subprogram</td>
<td>24</td>
<td>M</td>
<td>0</td>
<td>-1 5)</td>
<td>1</td>
</tr>
<tr>
<td>Nibbling</td>
<td>25</td>
<td>M</td>
<td>0</td>
<td>20 6)</td>
<td>(10)</td>
</tr>
<tr>
<td>Nibbling</td>
<td>26</td>
<td>M</td>
<td>0</td>
<td>23 6)</td>
<td>(10)</td>
</tr>
<tr>
<td>Nibbling</td>
<td>27</td>
<td>M</td>
<td>0</td>
<td>22 6)</td>
<td>(11)</td>
</tr>
<tr>
<td>Nibbling</td>
<td>28</td>
<td>M</td>
<td>0</td>
<td>25 6)</td>
<td>(11)</td>
</tr>
<tr>
<td>Nibbling</td>
<td>29</td>
<td>M</td>
<td>0</td>
<td>26 6)</td>
<td>(12)</td>
</tr>
<tr>
<td>Nibbling</td>
<td>30</td>
<td>M</td>
<td>0</td>
<td>122 6)</td>
<td>(11)</td>
</tr>
<tr>
<td>Nibbling</td>
<td>31</td>
<td>M</td>
<td>0</td>
<td>125 6)</td>
<td>(11)</td>
</tr>
<tr>
<td>Nibbling</td>
<td>32</td>
<td>M</td>
<td>0</td>
<td>27 6)</td>
<td>(12)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 2

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>33</td>
<td>M</td>
<td>2</td>
<td>3</td>
<td>(72)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>34</td>
<td>M</td>
<td>2</td>
<td>4</td>
<td>(72)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>35</td>
<td>M</td>
<td>2</td>
<td>5</td>
<td>(72)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>36</td>
<td>M</td>
<td>2</td>
<td>19</td>
<td>(72)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>37</td>
<td>M</td>
<td>2</td>
<td>70 2)</td>
<td>(72)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>38</td>
<td>M</td>
<td>2</td>
<td>40</td>
<td>(74)</td>
</tr>
</tbody>
</table>
### Spindle-specific auxiliary functions, spindle 2

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gear stage 1</td>
<td>39</td>
<td>M</td>
<td>2</td>
<td>41</td>
<td>(74)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>40</td>
<td>M</td>
<td>2</td>
<td>42</td>
<td>(74)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>41</td>
<td>M</td>
<td>2</td>
<td>43</td>
<td>(74)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>42</td>
<td>M</td>
<td>2</td>
<td>44</td>
<td>(74)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>43</td>
<td>M</td>
<td>2</td>
<td>45</td>
<td>(74)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>44</td>
<td>S</td>
<td>2</td>
<td>-1</td>
<td>(73)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 3

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>45</td>
<td>M</td>
<td>3</td>
<td>3</td>
<td>(75)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>46</td>
<td>M</td>
<td>3</td>
<td>4</td>
<td>(75)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>47</td>
<td>M</td>
<td>3</td>
<td>5</td>
<td>(75)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>48</td>
<td>M</td>
<td>3</td>
<td>19</td>
<td>(75)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>49</td>
<td>M</td>
<td>3</td>
<td>70 (^2)</td>
<td>(75)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>50</td>
<td>M</td>
<td>3</td>
<td>40</td>
<td>(77)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>51</td>
<td>M</td>
<td>3</td>
<td>41</td>
<td>(77)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>52</td>
<td>M</td>
<td>3</td>
<td>42</td>
<td>(77)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>53</td>
<td>M</td>
<td>3</td>
<td>43</td>
<td>(77)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>54</td>
<td>M</td>
<td>3</td>
<td>44</td>
<td>(77)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>55</td>
<td>M</td>
<td>3</td>
<td>45</td>
<td>(77)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>56</td>
<td>S</td>
<td>3</td>
<td>-1</td>
<td>(76)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 4

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>57</td>
<td>M</td>
<td>4</td>
<td>3</td>
<td>(78)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>58</td>
<td>M</td>
<td>4</td>
<td>4</td>
<td>(78)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>59</td>
<td>M</td>
<td>4</td>
<td>5</td>
<td>(78)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>60</td>
<td>M</td>
<td>4</td>
<td>19</td>
<td>(78)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>61</td>
<td>M</td>
<td>4</td>
<td>70 (^2)</td>
<td>(78)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>62</td>
<td>M</td>
<td>4</td>
<td>40</td>
<td>(80)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>63</td>
<td>M</td>
<td>4</td>
<td>41</td>
<td>(80)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>64</td>
<td>M</td>
<td>4</td>
<td>42</td>
<td>(80)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>65</td>
<td>M</td>
<td>4</td>
<td>43</td>
<td>(80)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>66</td>
<td>M</td>
<td>4</td>
<td>44</td>
<td>(80)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>67</td>
<td>M</td>
<td>4</td>
<td>45</td>
<td>(80)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>68</td>
<td>S</td>
<td>4</td>
<td>-1</td>
<td>(79)</td>
</tr>
</tbody>
</table>
### 8.2 Predefined auxiliary functions

#### Spindle-specific auxiliary functions, spindle 5

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>69</td>
<td>M</td>
<td>5</td>
<td>3</td>
<td>(81)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>70</td>
<td>M</td>
<td>5</td>
<td>4</td>
<td>(81)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>71</td>
<td>M</td>
<td>5</td>
<td>5</td>
<td>(81)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>72</td>
<td>M</td>
<td>5</td>
<td>19</td>
<td>(81)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>73</td>
<td>M</td>
<td>5</td>
<td>70 ²</td>
<td>(81)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>74</td>
<td>M</td>
<td>5</td>
<td>40</td>
<td>(83)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>75</td>
<td>M</td>
<td>5</td>
<td>41</td>
<td>(83)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>76</td>
<td>M</td>
<td>5</td>
<td>42</td>
<td>(83)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>77</td>
<td>M</td>
<td>5</td>
<td>43</td>
<td>(83)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>78</td>
<td>M</td>
<td>5</td>
<td>44</td>
<td>(83)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>79</td>
<td>M</td>
<td>5</td>
<td>45</td>
<td>(83)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>80</td>
<td>S</td>
<td>5</td>
<td>-1</td>
<td>(82)</td>
</tr>
</tbody>
</table>

#### Spindle-specific auxiliary functions, spindle 6

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>81</td>
<td>M</td>
<td>6</td>
<td>3</td>
<td>(84)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>82</td>
<td>M</td>
<td>6</td>
<td>4</td>
<td>(84)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>83</td>
<td>M</td>
<td>6</td>
<td>5</td>
<td>(84)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>84</td>
<td>M</td>
<td>6</td>
<td>19</td>
<td>(84)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>85</td>
<td>M</td>
<td>6</td>
<td>70 ²</td>
<td>(84)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>86</td>
<td>M</td>
<td>6</td>
<td>40</td>
<td>(86)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>87</td>
<td>M</td>
<td>6</td>
<td>41</td>
<td>(86)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>88</td>
<td>M</td>
<td>6</td>
<td>42</td>
<td>(86)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>89</td>
<td>M</td>
<td>6</td>
<td>43</td>
<td>(86)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>90</td>
<td>M</td>
<td>6</td>
<td>44</td>
<td>(86)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>91</td>
<td>M</td>
<td>6</td>
<td>45</td>
<td>(86)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>92</td>
<td>S</td>
<td>6</td>
<td>-1</td>
<td>(85)</td>
</tr>
</tbody>
</table>

#### Spindle-specific auxiliary functions, spindle 7

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>93</td>
<td>M</td>
<td>7</td>
<td>3</td>
<td>(87)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>94</td>
<td>M</td>
<td>7</td>
<td>4</td>
<td>(87)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>95</td>
<td>M</td>
<td>7</td>
<td>5</td>
<td>(87)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>96</td>
<td>M</td>
<td>7</td>
<td>19</td>
<td>(87)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>97</td>
<td>M</td>
<td>7</td>
<td>70 ²</td>
<td>(87)</td>
</tr>
</tbody>
</table>

2) Additional notes or explanations related to these values are not provided in the table but are typically included in the manual for reference.
### Spindle-specific auxiliary functions, spindle 7

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic gear stage</td>
<td>98</td>
<td>M</td>
<td>7</td>
<td>40</td>
<td>(89)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>99</td>
<td>M</td>
<td>7</td>
<td>41</td>
<td>(89)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>100</td>
<td>M</td>
<td>7</td>
<td>42</td>
<td>(89)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>101</td>
<td>M</td>
<td>7</td>
<td>43</td>
<td>(89)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>102</td>
<td>M</td>
<td>7</td>
<td>44</td>
<td>(89)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>103</td>
<td>M</td>
<td>7</td>
<td>45</td>
<td>(89)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>104</td>
<td>S</td>
<td>7</td>
<td>-1</td>
<td>(88)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 8

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>105</td>
<td>M</td>
<td>8</td>
<td>3</td>
<td>(90)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>106</td>
<td>M</td>
<td>8</td>
<td>4</td>
<td>(90)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>107</td>
<td>M</td>
<td>8</td>
<td>5</td>
<td>(90)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>108</td>
<td>M</td>
<td>8</td>
<td>19</td>
<td>(90)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>109</td>
<td>M</td>
<td>8</td>
<td>70&lt;sup&gt;2&lt;/sup&gt;</td>
<td>(90)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>110</td>
<td>M</td>
<td>8</td>
<td>40</td>
<td>(92)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>111</td>
<td>M</td>
<td>8</td>
<td>41</td>
<td>(92)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>112</td>
<td>M</td>
<td>8</td>
<td>42</td>
<td>(92)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>113</td>
<td>M</td>
<td>8</td>
<td>43</td>
<td>(92)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>114</td>
<td>M</td>
<td>8</td>
<td>44</td>
<td>(92)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>115</td>
<td>M</td>
<td>8</td>
<td>45</td>
<td>(92)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>116</td>
<td>S</td>
<td>8</td>
<td>-1</td>
<td>(91)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 9

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>117</td>
<td>M</td>
<td>9</td>
<td>3</td>
<td>(93)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>118</td>
<td>M</td>
<td>9</td>
<td>4</td>
<td>(93)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>119</td>
<td>M</td>
<td>9</td>
<td>5</td>
<td>(93)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>120</td>
<td>M</td>
<td>9</td>
<td>19</td>
<td>(93)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>121</td>
<td>M</td>
<td>9</td>
<td>70&lt;sup&gt;2&lt;/sup&gt;</td>
<td>(93)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>122</td>
<td>M</td>
<td>9</td>
<td>40</td>
<td>(95)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>123</td>
<td>M</td>
<td>9</td>
<td>41</td>
<td>(95)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>124</td>
<td>M</td>
<td>9</td>
<td>42</td>
<td>(95)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>125</td>
<td>M</td>
<td>9</td>
<td>43</td>
<td>(95)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>126</td>
<td>M</td>
<td>9</td>
<td>44</td>
<td>(95)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>127</td>
<td>M</td>
<td>9</td>
<td>45</td>
<td>(95)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>128</td>
<td>S</td>
<td>9</td>
<td>-1</td>
<td>(94)</td>
</tr>
</tbody>
</table>
### Spindle-specific auxiliary functions, spindle 10

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>129</td>
<td>M</td>
<td>10</td>
<td>3</td>
<td>(96)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>130</td>
<td>M</td>
<td>10</td>
<td>4</td>
<td>(96)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>131</td>
<td>M</td>
<td>10</td>
<td>5</td>
<td>(96)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>132</td>
<td>M</td>
<td>10</td>
<td>19</td>
<td>(96)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>133</td>
<td>M</td>
<td>10</td>
<td>70(^2)</td>
<td>(96)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>134</td>
<td>M</td>
<td>10</td>
<td>40</td>
<td>(98)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>135</td>
<td>M</td>
<td>10</td>
<td>41</td>
<td>(98)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>136</td>
<td>M</td>
<td>10</td>
<td>42</td>
<td>(98)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>137</td>
<td>M</td>
<td>10</td>
<td>43</td>
<td>(98)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>138</td>
<td>M</td>
<td>10</td>
<td>44</td>
<td>(98)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>139</td>
<td>M</td>
<td>10</td>
<td>45</td>
<td>(98)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>140</td>
<td>S</td>
<td>10</td>
<td>-1</td>
<td>(97)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 11

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>141</td>
<td>M</td>
<td>11</td>
<td>3</td>
<td>(99)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>142</td>
<td>M</td>
<td>11</td>
<td>4</td>
<td>(99)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>143</td>
<td>M</td>
<td>11</td>
<td>5</td>
<td>(99)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>144</td>
<td>M</td>
<td>11</td>
<td>19</td>
<td>(99)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>145</td>
<td>M</td>
<td>11</td>
<td>70(^2)</td>
<td>(99)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>146</td>
<td>M</td>
<td>11</td>
<td>40</td>
<td>(101)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>147</td>
<td>M</td>
<td>11</td>
<td>41</td>
<td>(101)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>148</td>
<td>M</td>
<td>11</td>
<td>42</td>
<td>(101)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>149</td>
<td>M</td>
<td>11</td>
<td>43</td>
<td>(101)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>150</td>
<td>M</td>
<td>11</td>
<td>44</td>
<td>(101)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>151</td>
<td>M</td>
<td>11</td>
<td>45</td>
<td>(101)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>152</td>
<td>S</td>
<td>11</td>
<td>-1</td>
<td>(100)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 12

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>153</td>
<td>M</td>
<td>11</td>
<td>3</td>
<td>(102)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>154</td>
<td>M</td>
<td>12</td>
<td>4</td>
<td>(102)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>155</td>
<td>M</td>
<td>12</td>
<td>5</td>
<td>(102)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>156</td>
<td>M</td>
<td>12</td>
<td>19</td>
<td>(102)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>157</td>
<td>M</td>
<td>12</td>
<td>70(^2)</td>
<td>(102)</td>
</tr>
</tbody>
</table>
### Spindle-specific auxiliary functions, spindle 12

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic gear stage</td>
<td>158</td>
<td>M</td>
<td>12</td>
<td>40</td>
<td>(104)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>159</td>
<td>M</td>
<td>12</td>
<td>41</td>
<td>(104)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>160</td>
<td>M</td>
<td>12</td>
<td>42</td>
<td>(104)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>161</td>
<td>M</td>
<td>12</td>
<td>43</td>
<td>(104)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>162</td>
<td>M</td>
<td>12</td>
<td>44</td>
<td>(104)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>163</td>
<td>M</td>
<td>12</td>
<td>45</td>
<td>(104)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>164</td>
<td>S</td>
<td>12</td>
<td>-1</td>
<td>(103)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 13

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>165</td>
<td>M</td>
<td>13</td>
<td>3</td>
<td>(105)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>166</td>
<td>M</td>
<td>13</td>
<td>4</td>
<td>(105)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>167</td>
<td>M</td>
<td>13</td>
<td>5</td>
<td>(105)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>168</td>
<td>M</td>
<td>13</td>
<td>19</td>
<td>(105)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>169</td>
<td>M</td>
<td>13</td>
<td>70 2)</td>
<td>(105)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>170</td>
<td>M</td>
<td>13</td>
<td>40</td>
<td>(107)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>171</td>
<td>M</td>
<td>13</td>
<td>41</td>
<td>(107)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>172</td>
<td>M</td>
<td>13</td>
<td>42</td>
<td>(107)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>173</td>
<td>M</td>
<td>13</td>
<td>43</td>
<td>(107)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>174</td>
<td>M</td>
<td>13</td>
<td>44</td>
<td>(107)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>175</td>
<td>M</td>
<td>13</td>
<td>45</td>
<td>(107)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>176</td>
<td>S</td>
<td>13</td>
<td>-1</td>
<td>(106)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 14

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>177</td>
<td>M</td>
<td>14</td>
<td>3</td>
<td>(108)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>178</td>
<td>M</td>
<td>14</td>
<td>4</td>
<td>(108)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>179</td>
<td>M</td>
<td>14</td>
<td>5</td>
<td>(108)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>180</td>
<td>M</td>
<td>14</td>
<td>19</td>
<td>(108)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>181</td>
<td>M</td>
<td>14</td>
<td>70 2)</td>
<td>(108)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>182</td>
<td>M</td>
<td>14</td>
<td>40</td>
<td>(110)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>183</td>
<td>M</td>
<td>14</td>
<td>41</td>
<td>(110)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>184</td>
<td>M</td>
<td>14</td>
<td>42</td>
<td>(110)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>185</td>
<td>M</td>
<td>14</td>
<td>43</td>
<td>(110)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>186</td>
<td>M</td>
<td>14</td>
<td>44</td>
<td>(110)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>187</td>
<td>M</td>
<td>14</td>
<td>45</td>
<td>(110)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>188</td>
<td>S</td>
<td>14</td>
<td>-1</td>
<td>(109)</td>
</tr>
</tbody>
</table>
### Spindle-specific auxiliary functions, spindle 15

<table>
<thead>
<tr>
<th>System function</th>
<th>Index (n)</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>189</td>
<td>M</td>
<td>15</td>
<td>3</td>
<td>(111)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>190</td>
<td>M</td>
<td>15</td>
<td>4</td>
<td>(111)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>191</td>
<td>M</td>
<td>15</td>
<td>5</td>
<td>(111)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>192</td>
<td>M</td>
<td>15</td>
<td>19</td>
<td>(111)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>193</td>
<td>M</td>
<td>15</td>
<td>70 (^2)</td>
<td>(111)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>194</td>
<td>M</td>
<td>15</td>
<td>40</td>
<td>(113)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>195</td>
<td>M</td>
<td>15</td>
<td>41</td>
<td>(113)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>196</td>
<td>M</td>
<td>15</td>
<td>42</td>
<td>(113)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>197</td>
<td>M</td>
<td>15</td>
<td>43</td>
<td>(113)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>198</td>
<td>M</td>
<td>15</td>
<td>44</td>
<td>(113)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>199</td>
<td>M</td>
<td>15</td>
<td>45</td>
<td>(113)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>200</td>
<td>S</td>
<td>15</td>
<td>-1</td>
<td>(112)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 16

<table>
<thead>
<tr>
<th>System function</th>
<th>Index (n)</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>201</td>
<td>M</td>
<td>16</td>
<td>3</td>
<td>(114)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>202</td>
<td>M</td>
<td>16</td>
<td>4</td>
<td>(114)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>203</td>
<td>M</td>
<td>16</td>
<td>5</td>
<td>(114)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>204</td>
<td>M</td>
<td>16</td>
<td>19</td>
<td>(114)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>205</td>
<td>M</td>
<td>16</td>
<td>70 (^2)</td>
<td>(114)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>206</td>
<td>M</td>
<td>16</td>
<td>40</td>
<td>(116)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>207</td>
<td>M</td>
<td>16</td>
<td>41</td>
<td>(116)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>208</td>
<td>M</td>
<td>16</td>
<td>42</td>
<td>(116)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>209</td>
<td>M</td>
<td>16</td>
<td>43</td>
<td>(116)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>210</td>
<td>M</td>
<td>16</td>
<td>44</td>
<td>(116)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>211</td>
<td>M</td>
<td>16</td>
<td>45</td>
<td>(116)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>212</td>
<td>S</td>
<td>16</td>
<td>-1</td>
<td>(115)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 17

<table>
<thead>
<tr>
<th>System function</th>
<th>Index (n)</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>213</td>
<td>M</td>
<td>17</td>
<td>3</td>
<td>(117)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>214</td>
<td>M</td>
<td>17</td>
<td>4</td>
<td>(117)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>215</td>
<td>M</td>
<td>17</td>
<td>5</td>
<td>(117)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>216</td>
<td>M</td>
<td>17</td>
<td>19</td>
<td>(117)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>217</td>
<td>M</td>
<td>17</td>
<td>70 (^2)</td>
<td>(117)</td>
</tr>
</tbody>
</table>

---

**H2: Auxiliary function outputs to PLC**

8.2 Predefined auxiliary functions
### Spindle-specific auxiliary functions, spindle 17

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic gear stage</td>
<td>218</td>
<td>M</td>
<td>17</td>
<td>40</td>
<td>(119)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>219</td>
<td>M</td>
<td>17</td>
<td>41</td>
<td>(119)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>220</td>
<td>M</td>
<td>17</td>
<td>42</td>
<td>(119)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>221</td>
<td>M</td>
<td>17</td>
<td>43</td>
<td>(119)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>222</td>
<td>M</td>
<td>17</td>
<td>44</td>
<td>(119)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>223</td>
<td>M</td>
<td>17</td>
<td>45</td>
<td>(119)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>224</td>
<td>S</td>
<td>17</td>
<td>-1</td>
<td>(118)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 18

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>225</td>
<td>M</td>
<td>18</td>
<td>3</td>
<td>(120)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>226</td>
<td>M</td>
<td>18</td>
<td>4</td>
<td>(120)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>227</td>
<td>M</td>
<td>18</td>
<td>5</td>
<td>(120)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>228</td>
<td>M</td>
<td>18</td>
<td>19</td>
<td>(120)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>229</td>
<td>M</td>
<td>18</td>
<td>70 2)</td>
<td>(120)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>230</td>
<td>M</td>
<td>18</td>
<td>40</td>
<td>(122)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>231</td>
<td>M</td>
<td>18</td>
<td>41</td>
<td>(122)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>232</td>
<td>M</td>
<td>18</td>
<td>42</td>
<td>(122)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>233</td>
<td>M</td>
<td>18</td>
<td>43</td>
<td>(122)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>234</td>
<td>M</td>
<td>18</td>
<td>44</td>
<td>(122)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>235</td>
<td>M</td>
<td>18</td>
<td>45</td>
<td>(122)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>236</td>
<td>S</td>
<td>18</td>
<td>-1</td>
<td>(121)</td>
</tr>
</tbody>
</table>

### Spindle-specific auxiliary functions, spindle 19

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>237</td>
<td>M</td>
<td>19</td>
<td>3</td>
<td>(123)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>238</td>
<td>M</td>
<td>19</td>
<td>4</td>
<td>(123)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>239</td>
<td>M</td>
<td>19</td>
<td>5</td>
<td>(123)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>240</td>
<td>M</td>
<td>19</td>
<td>19</td>
<td>(123)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>241</td>
<td>M</td>
<td>19</td>
<td>70 2)</td>
<td>(123)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>242</td>
<td>M</td>
<td>19</td>
<td>40</td>
<td>(125)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>243</td>
<td>M</td>
<td>19</td>
<td>41</td>
<td>(125)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>244</td>
<td>M</td>
<td>19</td>
<td>42</td>
<td>(125)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>245</td>
<td>M</td>
<td>19</td>
<td>43</td>
<td>(125)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>246</td>
<td>M</td>
<td>19</td>
<td>44</td>
<td>(125)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>247</td>
<td>M</td>
<td>19</td>
<td>45</td>
<td>(125)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>248</td>
<td>S</td>
<td>19</td>
<td>-1</td>
<td>(124)</td>
</tr>
</tbody>
</table>
### Spindle-specific auxiliary functions, spindle 20

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle right</td>
<td>249</td>
<td>M</td>
<td>20</td>
<td>3</td>
<td>(126)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>250</td>
<td>M</td>
<td>20</td>
<td>4</td>
<td>(126)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>251</td>
<td>M</td>
<td>20</td>
<td>5</td>
<td>(126)</td>
</tr>
<tr>
<td>Position spindle</td>
<td>252</td>
<td>M</td>
<td>20</td>
<td>19</td>
<td>(126)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>253</td>
<td>M</td>
<td>20</td>
<td>70</td>
<td>(126)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>254</td>
<td>M</td>
<td>20</td>
<td>40</td>
<td>(128)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>255</td>
<td>M</td>
<td>20</td>
<td>41</td>
<td>(128)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>256</td>
<td>M</td>
<td>20</td>
<td>42</td>
<td>(128)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>257</td>
<td>M</td>
<td>20</td>
<td>43</td>
<td>(128)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>258</td>
<td>M</td>
<td>20</td>
<td>44</td>
<td>(128)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>259</td>
<td>M</td>
<td>20</td>
<td>45</td>
<td>(128)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>260</td>
<td>S</td>
<td>20</td>
<td>-1</td>
<td>(127)</td>
</tr>
</tbody>
</table>

### Toolholder-specific auxiliary functions, T auxiliary functions

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool selection</td>
<td>261</td>
<td>T</td>
<td>1</td>
<td>-1</td>
<td>129</td>
</tr>
<tr>
<td>Tool selection</td>
<td>262</td>
<td>T</td>
<td>2</td>
<td>-1</td>
<td>130</td>
</tr>
<tr>
<td>Tool selection</td>
<td>263</td>
<td>T</td>
<td>3</td>
<td>-1</td>
<td>131</td>
</tr>
<tr>
<td>Tool selection</td>
<td>264</td>
<td>T</td>
<td>4</td>
<td>-1</td>
<td>132</td>
</tr>
<tr>
<td>Tool selection</td>
<td>265</td>
<td>T</td>
<td>5</td>
<td>-1</td>
<td>133</td>
</tr>
<tr>
<td>Tool selection</td>
<td>266</td>
<td>T</td>
<td>6</td>
<td>-1</td>
<td>134</td>
</tr>
<tr>
<td>Tool selection</td>
<td>267</td>
<td>T</td>
<td>7</td>
<td>-1</td>
<td>135</td>
</tr>
<tr>
<td>Tool selection</td>
<td>268</td>
<td>T</td>
<td>8</td>
<td>-1</td>
<td>136</td>
</tr>
<tr>
<td>Tool selection</td>
<td>269</td>
<td>T</td>
<td>9</td>
<td>-1</td>
<td>137</td>
</tr>
<tr>
<td>Tool selection</td>
<td>270</td>
<td>T</td>
<td>10</td>
<td>-1</td>
<td>138</td>
</tr>
<tr>
<td>Tool selection</td>
<td>271</td>
<td>T</td>
<td>11</td>
<td>-1</td>
<td>139</td>
</tr>
<tr>
<td>Tool selection</td>
<td>272</td>
<td>T</td>
<td>12</td>
<td>-1</td>
<td>140</td>
</tr>
<tr>
<td>Tool selection</td>
<td>273</td>
<td>T</td>
<td>13</td>
<td>-1</td>
<td>141</td>
</tr>
<tr>
<td>Tool selection</td>
<td>274</td>
<td>T</td>
<td>14</td>
<td>-1</td>
<td>142</td>
</tr>
<tr>
<td>Tool selection</td>
<td>275</td>
<td>T</td>
<td>15</td>
<td>-1</td>
<td>143</td>
</tr>
<tr>
<td>Tool selection</td>
<td>276</td>
<td>T</td>
<td>16</td>
<td>-1</td>
<td>144</td>
</tr>
<tr>
<td>Tool selection</td>
<td>277</td>
<td>T</td>
<td>17</td>
<td>-1</td>
<td>145</td>
</tr>
<tr>
<td>Tool selection</td>
<td>278</td>
<td>T</td>
<td>18</td>
<td>-1</td>
<td>146</td>
</tr>
<tr>
<td>Tool selection</td>
<td>279</td>
<td>T</td>
<td>19</td>
<td>-1</td>
<td>147</td>
</tr>
<tr>
<td>Tool selection</td>
<td>280</td>
<td>T</td>
<td>20</td>
<td>-1</td>
<td>148</td>
</tr>
</tbody>
</table>
### Toolholder-specific auxiliary functions, M6 auxiliary functions

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool change</td>
<td>281</td>
<td>M</td>
<td>1</td>
<td>6(^1)</td>
<td>149</td>
</tr>
<tr>
<td>Tool change</td>
<td>282</td>
<td>M</td>
<td>2</td>
<td>6(^1)</td>
<td>150</td>
</tr>
<tr>
<td>Tool change</td>
<td>283</td>
<td>M</td>
<td>3</td>
<td>6(^1)</td>
<td>151</td>
</tr>
<tr>
<td>Tool change</td>
<td>284</td>
<td>M</td>
<td>4</td>
<td>6(^1)</td>
<td>152</td>
</tr>
<tr>
<td>Tool change</td>
<td>285</td>
<td>M</td>
<td>5</td>
<td>6(^1)</td>
<td>153</td>
</tr>
<tr>
<td>Tool change</td>
<td>286</td>
<td>M</td>
<td>6</td>
<td>6(^1)</td>
<td>154</td>
</tr>
<tr>
<td>Tool change</td>
<td>287</td>
<td>M</td>
<td>7</td>
<td>6(^1)</td>
<td>155</td>
</tr>
<tr>
<td>Tool change</td>
<td>288</td>
<td>M</td>
<td>8</td>
<td>6(^1)</td>
<td>156</td>
</tr>
<tr>
<td>Tool change</td>
<td>289</td>
<td>M</td>
<td>9</td>
<td>6(^1)</td>
<td>157</td>
</tr>
<tr>
<td>Tool change</td>
<td>290</td>
<td>M</td>
<td>10</td>
<td>6(^1)</td>
<td>158</td>
</tr>
<tr>
<td>Tool change</td>
<td>291</td>
<td>M</td>
<td>11</td>
<td>6(^1)</td>
<td>159</td>
</tr>
<tr>
<td>Tool change</td>
<td>292</td>
<td>M</td>
<td>12</td>
<td>6(^1)</td>
<td>160</td>
</tr>
<tr>
<td>Tool change</td>
<td>293</td>
<td>M</td>
<td>13</td>
<td>6(^1)</td>
<td>161</td>
</tr>
<tr>
<td>Tool change</td>
<td>294</td>
<td>M</td>
<td>14</td>
<td>6(^1)</td>
<td>162</td>
</tr>
<tr>
<td>Tool change</td>
<td>295</td>
<td>M</td>
<td>15</td>
<td>6(^1)</td>
<td>163</td>
</tr>
<tr>
<td>Tool change</td>
<td>296</td>
<td>M</td>
<td>16</td>
<td>6(^1)</td>
<td>164</td>
</tr>
<tr>
<td>Tool change</td>
<td>297</td>
<td>M</td>
<td>17</td>
<td>6(^1)</td>
<td>165</td>
</tr>
<tr>
<td>Tool change</td>
<td>298</td>
<td>M</td>
<td>18</td>
<td>6(^1)</td>
<td>166</td>
</tr>
<tr>
<td>Tool change</td>
<td>299</td>
<td>M</td>
<td>19</td>
<td>6(^1)</td>
<td>167</td>
</tr>
<tr>
<td>Tool change</td>
<td>300</td>
<td>M</td>
<td>20</td>
<td>6(^1)</td>
<td>168</td>
</tr>
</tbody>
</table>

**Legend:**

\(\) The value can be changed.

\(^1\) The value is depends on the machine data:

MD22560 $MC_TOOL_CHANGE_M_MODE (M function for tool change)

\(^2\) The value can be preset with a different value using the following machine data:

MD20095 $MC_EXTERN_RIGID_TAPPING_M_NR (M function for switching over to controlled axis mode (ext. mode))

MD20094 $MC_SPIND_RIGID_TAPPING_M_NR (M function for switching over to controlled axis mode)

Note
The value 70 is always output at the PLC.

\(^3\) The value is set using machine data:

MD22254 $MC_AUXFU_ASSOC_M0_VALUE (additional M function for program stop)

\(^4\) The value is set using machine data:

MD22256 $MC_AUXFU_ASSOC_M1_VALUE (additional M function for conditional stop)
8.2 Overview: Output behavior

Significance of the parameters listed in the following table:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index &lt;n&gt;</td>
<td>Machine data index of the parameters of an auxiliary function</td>
</tr>
<tr>
<td>Output behavior</td>
<td>MD22080 $MC_AUXFU_PREDEF_SPEC[&lt;n&gt;], Bits 0 ... 18</td>
</tr>
<tr>
<td></td>
<td>Bits 19 ... 31: Reserved</td>
</tr>
</tbody>
</table>

Output behavior of the predefined auxiliary functions

<table>
<thead>
<tr>
<th>System function</th>
<th>Index &lt;n&gt;</th>
<th>Output behavior, bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>Stop</td>
<td>0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Conditional stop</td>
<td>1 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>End of subroutine</td>
<td>2 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Tool change</td>
<td>5 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Spindle right</td>
<td>6 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Spindle left</td>
<td>7 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>8 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Spindle positioning</td>
<td>9 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Axis mode</td>
<td>10 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>11 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>12 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>13 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>14 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>15 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>16 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>17 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Feed</td>
<td>18 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Cutting edge selection</td>
<td>19 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>DL</td>
<td>20 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Tool selection</td>
<td>21 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
<tr>
<td>Stop (associated)</td>
<td>22 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (0) (1)</td>
</tr>
</tbody>
</table>
### System function

| Conditional stop (associated) | 23 | 0 | 0 | 0 | (0) | 0 | 0 | 0 | 0 | (0) | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | (0) | (1) |
|-------------------------------|----|---|---|---|-----|---|---|---|---|-----|---|---|---|---|---|---|---|---|---|---|----|
| End of subroutine             | 24 | 0 | 0 | 0 | (0) | 0 | 0 | 0 | 0 | (0) | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | (0) | (1) |
| Nibbling                      | 25 | 0 | 0 | 0 | (0) | 0 | 0 | 0 | 0 | (0) | 0 | 0 | 1 | 0 | 0 | (0) | (0) | (0) | (0) | (1) |
| Nibbling                      | 26 | 0 | 0 | 0 | (0) | 0 | 0 | 0 | 0 | (0) | 0 | 0 | 1 | 0 | 0 | (0) | (0) | (0) | (0) | (1) |
| Nibbling                      | 27 | 0 | 0 | 0 | (0) | 0 | 0 | 0 | 0 | (0) | 0 | 0 | 1 | 0 | 0 | (0) | (0) | (0) | (0) | (1) |
| Nibbling                      | 28 | 0 | 0 | 0 | (0) | 0 | 0 | 0 | 0 | (0) | 0 | 0 | 1 | 0 | 0 | (0) | (0) | (0) | (1) | (0) |
| Nibbling                      | 29 | 0 | 0 | 0 | (0) | 0 | 0 | 0 | 0 | (0) | 0 | 0 | 1 | 0 | 0 | (0) | (0) | (0) | (1) | (0) |
| Nibbling                      | 30 | 0 | 0 | 0 | (0) | 0 | 0 | 0 | 0 | (0) | 0 | 0 | 1 | 0 | 0 | (0) | (0) | (0) | (1) | (0) |
| Nibbling                      | 31 | 0 | 0 | 0 | (0) | 0 | 0 | 0 | 0 | (0) | 0 | 0 | 1 | 0 | 0 | (0) | (0) | (0) | (1) | (0) |
| Nibbling                      | 32 | 0 | 0 | 0 | (0) | 0 | 0 | 0 | 0 | (0) | 0 | 0 | 1 | 0 | 0 | (0) | (0) | (0) | (1) | (0) |

The value can be changed.

### Significance of the bits

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Acknowledgement &quot;normal&quot; after an OB1 cycle</td>
</tr>
<tr>
<td></td>
<td>An auxiliary function with normal acknowledgment is output to the NC/PLC interface at the beginning of the OB1 cycle. The auxiliary function-specific change signal indicates to the PLC user program that the auxiliary function is valid. The auxiliary function is acknowledged as soon as organization block OB1 has run once. This corresponds to a complete PLC user cycle. The auxiliary function with normal acknowledgment is output in synchronism with the part program block in which it is programmed. If execution of the parts program block, e.g. path and/or positioning axis movements, is completed before acknowledgment of the auxiliary function, the block change is delayed until after acknowledgment by the PLC. In continuous-path mode, a constant path velocity can be maintained in conjunction with an auxiliary function with normal acknowledgment, if the auxiliary function is output by the PLC during the traversing motion and before reaching the end of the block.</td>
</tr>
<tr>
<td>1</td>
<td>Acknowledgement &quot;quick&quot; with OB40</td>
</tr>
<tr>
<td></td>
<td>An auxiliary function with quick acknowledgment is output to the NC/PLC interface before the next OB1 cycle. The auxiliary function-specific change signal indicates to the PLC user program that the auxiliary function is valid. The auxiliary function is acknowledged immediately by the PLC basic program in the next OB40 cycle. Acknowledgment of the auxiliary function is not confirmation that the corresponding PLC user function has been executed. The auxiliary function is still executed in the OB1 cycle. Next output of the auxiliary functions to the PLC is therefore not possible until after this OB1 cycle has run completely. This is noticeable in continuous-path mode (drop in path velocity) especially if auxiliary functions with quick acknowledgment are output in several consecutive part program blocks. With auxiliary functions with quick acknowledgment, it cannot be guaranteed that the PLC user program will respond in synchronism with the block.</td>
</tr>
<tr>
<td></td>
<td>Note Parameterization of the output behavior of auxiliary functions as &quot;quick auxiliary functions&quot; is only possible in conjunction with user-defined auxiliary functions.</td>
</tr>
<tr>
<td>Bit</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>2</td>
<td>No predefined auxiliary function</td>
</tr>
</tbody>
</table>
|     | A predefined auxiliary function is treated like a user-defined auxiliary function with this setting. The auxiliary function then no longer triggers the corresponding system function but is only output to the PLC.  
|     | Example:  
|     | Reconfiguration of the "Position spindle" auxiliary function (index 9) to a user-defined auxiliary function with normal acknowledgment and output prior to the traversing motion.  
|     | MD22080 $MC_AUXFU_PREDEF_SPEC [ 9 ] = 'H25' (100101B) |
| 3   | No output to the PLC |
|     | The auxiliary function is not output to the PLC. |
| 4   | Spindle response after acknowledgement by the PLC |
|     | The associated system function is only executed after acknowledgment by the PLC. |
| 5   | Output prior to motion |
|     | The auxiliary function is output to the PLC before the traversing motions programmed in the part program block (path and/or block-related positioning axis movements). |
| 6   | Output during motion |
|     | The auxiliary function is output to the PLC during the traversing motions programmed in the part program block (path and/or block-related positioning axis movements). |
| 7   | Output at block end |
|     | The auxiliary function is output to the PLC after the traversing motions programmed in the part program block have been completed (path and/or block-related positioning axis movements). |
| 8   | Not output after block search, types 1, 2, 4 |
|     | Block search, types 1, 2, 4: The auxiliary function collected during the block search is not output. |
| 9   | Collection during block search with program test (type 5, SERUPRO) |
|     | For a block search with program test, the auxiliary function is collected group-specific in the following system variables:  
|     | ● $AC_AUXFU_M_VALUE[<n>]  
|     | ● $AC_AUXFU_M_EXT[<n>]  
|     | ● $AC_AUXFU_M_STATE[<n>] |
| 10  | No output during block search with program test (type 5, SERUPRO) |
|     | For block search with program test, the auxiliary function is not output to the PLC. |
| 11  | Cross-channel auxiliary function (SERUPRO) |
|     | For block search with program test (SERUPRO), the help function is collected on a cross-channel basis in the global list of the auxiliary functions.  
|     | Note  
|     | For each auxiliary function group, only the last auxiliary function of the group is always collected. |
| 12  | Output performed via synchronized action (read only) |
|     | The bit is set if the auxiliary function was output to the PLC via a synchronized action. |
| 13  | Implicit auxiliary function (read-only) |
|     | The bit is set if the auxiliary function was implicitly output to the PLC. |
| 14  | Active M01 (read only) |
|     | The bit is set if the auxiliary function, for active M01, was output to the PLC. |
| 15  | No output during positioning test run |
|     | During the run-in test, the auxiliary function is not output to the PLC. |
| 16  | Nibbling off |
### 8.2.3 Parameterization

#### 8.2.3.1 Group assignment

The handling of the auxiliary functions for a block search is defined using the group assignment of an auxiliary function. The 168 auxiliary function groups available are subdivided into predefined and user-definable groups:

- **Predefined groups:** 1 ... 4, 10 ... 12, 72 ... 168
- **User-defined groups:** 5 ... 9, 13 ... 71

Each predefined auxiliary function is assigned, as standard, to an auxiliary function group. For most pre-defined auxiliary functions, this assignment can be changed using the following machine data:

$$ \text{MD22040} \ $\text{MC_AUXFU_PREDEF_GROUP}[<n>] \ (\text{group assignment of predefined auxiliary functions}) $$

If an auxiliary function is not assigned to any group, then a value of "0" should be entered into the machine data.

For the pre-defined auxiliary functions with the following indices <n>, the group assignment cannot be changed: 0, 1, 2, 3, 4, 22, 23, 24

**Note**

1. **Auxiliary function group and block search**

Auxiliary functions of the 1st auxiliary function group are, for a block search, only collected, but not output.

#### 8.2.3.2 Type, address extension and value

An auxiliary function is programmed via the type, address extension and value parameters (see Section "Programming an auxiliary function (Page 412)").

---

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Nibbling on</td>
</tr>
<tr>
<td>18</td>
<td>Nibbling</td>
</tr>
</tbody>
</table>

**Note**

In the case of auxiliary functions for which no output behavior has been defined, the following default output behavior is active:

- Bit 0 = 1: Output duration one OB1 cycle
- Bit 7 = 1: Output at block end
### Type

The identifier of an auxiliary function is defined via the "type," e.g.:

| "M" | For additional function |
| "S" | For spindle function    |
| "F" | For feed                |

The setting is made via the following machine data:

MD22050 $MC_AUXFU_PREDEF_TYPE[<n>] (type of predefined auxiliary functions)

#### Note

The "type" cannot be changed for predefined auxiliary functions.

### Address extension

The "address extension" of an auxiliary function is for addressing different components of the same type. In the case of predefined auxiliary functions, the value of the "address extension" is the spindle number to which the auxiliary function applies.

The setting is made via the following machine data:

MD22060 $MC_AUXFU_PREDEF_EXTENSION[<n>] (address extension for predefined auxiliary functions)

#### Grouping together auxiliary functions

To assign an auxiliary function for all spindles of a channel to the same auxiliary function group, the value "-1" is entered for the "address extension" parameter.

Example:

The auxiliary function M3 (machine data index = 6) is assigned to the second auxiliary function group for all the channel's spindles.

MD22040 $MC_AUXFU_PREDEF_GROUP[6] = 2
MD22050 $MC_AUXFU_PREDEF_TYPE[6] = "M"
MD22060 $MC_AUXFU_PREDEF_EXTENSION[6] = -1
MD22070 $MC_AUXFU_PREDEF_VALUE[6] = 3

### Value

The parameters "value" and "type" define the meaning of an auxiliary function, i.e. the system function that is activated on the basis of this auxiliary function.

The "value" of an auxiliary function is defined in the machine data:
Note
The "value" cannot be changed for a predefined auxiliary function. For some predefined auxiliary functions, the "value" can be reconfigured via additional machine data (see Section "Associated auxiliary functions (Page 408)").

8.2.3.3 Output behavior

Parameter "Output behavior" defines when the predefined auxiliary function is output to the NC/PLC interface and when it is acknowledged by the PLC.

The setting is done via the following machine data:

MD22080 $MC_AUXFU_PREDEF_SPEC[<n>] (output behavior of predefined auxiliary functions)

Output behavior relative to motion

Output prior to motion
- The traversing motions (path and/or block-related positioning axis movements) of the previous part program block end with an exact stop.
- The auxiliary functions are output at the beginning of the current parts program block.
- The traversing motion of the actual part program block (path and/or positioning axis motion) is only started after acknowledgment of the auxiliary functions by the PLC:
  - Output duration one OB1 cycle (normal acknowledgment): after one OB1 cycle
  - Output duration one OB40 cycle (quick acknowledgment): after one OB40 cycle

Output during motion
- The auxiliary functions are output at the beginning of the traversing motions (path and/or positioning axis movements).
- The path velocity of the current parts program block is reduced so that the time to the end of the block is greater than the time to acknowledgment of the auxiliary functions by the PLC:
  - Output duration one OB1 cycle (normal acknowledgment): one OB1 cycle
  - Output duration one OB40 cycle (quick acknowledgment): one OB40 cycle

Output after motion
- The traversing motions (path and/or block-related positioning axis movements) of the current part program block end with an exact stop.
- The auxiliary functions are output after completion of the traversing motions.
- The block change is performed after acknowledgment of the auxiliary functions by the PLC:
  - Output duration one OB1 cycle (normal acknowledgment): after one OB1 cycle
  - Output duration one OB40 cycle (quick acknowledgment): after one OB40 cycle
Examples of different output behavior

The following figures illustrate the differing behavior regarding:
- Output and acknowledgment of the auxiliary function
- Spindle response (speed change)
- Traverse movement (velocity change)

The binary values specified in the diagrams under "Output behavior" refer to the parameterized output behavior (MD22080).
H2: Auxiliary function outputs to PLC

8.2 Predefined auxiliary functions
8.3 Userdefined auxiliary functions

There are two uses for user-defined auxiliary functions:

- Extension of predefined auxiliary functions
- User-specific auxiliary functions

Extension of predefined auxiliary functions

Because there is only one set of machine data for the predefined auxiliary functions, they can only ever be used to address one spindle of the channel. To address further spindles, user-defined auxiliary functions must be parameterized to supplement the predefined auxiliary functions.

Extension of predefined auxiliary functions refers to the "address extensions" parameter. The number of the spindle that the auxiliary function refers to is entered in the "address extension" parameter.
The relevant predefined auxiliary functions can be extended for the following system functions:

<table>
<thead>
<tr>
<th>System function</th>
<th>Type</th>
<th>Address extension</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool change</td>
<td>M</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Spindle right</td>
<td>M</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Spindle left</td>
<td>M</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Spindle stop</td>
<td>M</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Position spindle</td>
<td>M</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>Axis mode</td>
<td>M</td>
<td>1</td>
<td>70</td>
</tr>
<tr>
<td>Automatic gear stage</td>
<td>M</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>M</td>
<td>1</td>
<td>41</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>M</td>
<td>1</td>
<td>42</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>M</td>
<td>1</td>
<td>43</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>M</td>
<td>1</td>
<td>44</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>M</td>
<td>1</td>
<td>45</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>S</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>Tool selection</td>
<td>T</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>

1) Address extension = 1 is the default value used in the auxiliary functions predefined in the machine data

Example:

Extension of the predefined auxiliary function for the system function "spindle right" for the second and third spindle of the channel.

Auxiliary function "spindle right" for the second spindle of the channel:

```plaintext
MD22010 $MC_AUXFU_ASSIGN_TYPE[ n ] = "M"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION[ n ] = 2
MD22030 $MC_AUXFU.Assign_Value[ n ] = 3
```

Auxiliary function "spindle right" for the third spindle of the channel:

```plaintext
MD22010 $MC_AUXFU.Assign_Type[ m ] = "M"
MD22020 $MC_AUXFU.Assign_Extension[ m ] = 3
MD22030 $MC_AUXFU.Assign_Value[ m ] = 3
```

User-specific auxiliary functions

User-specific auxiliary functions have the following characteristics:

- User-specific auxiliary functions only activate user functions.
- No system functions can be activated by user-specific auxiliary functions.
A user-specific auxiliary function is output to the PLC according to the parameterized output behavior.

- The functionality of a user-specific auxiliary function is implemented by the machine manufacturer/user in the PLC user program.

### 8.3.1 Parameterization

#### 8.3.1.1 Maximum number of user-defined auxiliary functions

The maximum number of user-defined auxiliary function per channel can be parameterized via the machine data:

MD11100 $MN_AUXFU_MAXNUM_GROUP_ASSIGN (maximum number of user-defined auxiliary functions)

#### 8.3.1.2 Group assignment

The handling of the auxiliary functions for a block search is defined using the group assignment of an auxiliary function. The 168 auxiliary function groups available are subdivided into predefined and user-definable groups:

- **Predefined groups:**
  1 ... 4
  10 ... 12
  72 ... 168

- **User-defined groups:**
  5 ... 9
  13 ... 71

Every user-defined auxiliary function is assigned as standard to the 1st auxiliary function group. The assignment can be changed using the following machine data:

MD22000 $MC_AUXFU_ASSIGN_GROUP[<n>] (group assignment of user-defined auxiliary functions)

If an auxiliary function is not assigned to any group, then a value of "0" should be entered into the machine data.

**Note**

1. **Auxiliary function group and block search**

Auxiliary functions of the 1st auxiliary function group are, for a block search, only collected, but not output.

#### 8.3.1.3 Type, address extension and value

An auxiliary function is programmed via the type, address extension and value parameters (see Section "Programming an auxiliary function (Page 412)").
Type

The name of an auxiliary function is defined via the "type".

The identifiers for user-defined auxiliary functions are:

<table>
<thead>
<tr>
<th>Type</th>
<th>Identifier</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;H&quot;</td>
<td>Auxiliary function</td>
<td>User-specific auxiliary functions</td>
</tr>
<tr>
<td>&quot;M&quot;</td>
<td>Special function</td>
<td>Extension of predefined auxiliary functions</td>
</tr>
<tr>
<td>&quot;S&quot;</td>
<td>Spindle function</td>
<td></td>
</tr>
<tr>
<td>&quot;T&quot;</td>
<td>Tool number</td>
<td></td>
</tr>
</tbody>
</table>

The setting is made via the following machine data:

MD22010 $MC_AUXFU_ASSIGN_TYPE[<n>] (type of user-defined auxiliary functions)

Address extension

MD22020 $MC_AUXFU_ASSIGN_EXTENSION[<n>] (address extension user-defined auxiliary functions)

The functionality of the address extension is not defined in user-specific auxiliary functions. It is generally used to distinguish between auxiliary functions with the same "value".

Grouping together auxiliary functions

If all the auxiliary functions of the same type and address extension are assigned to the same auxiliary function group, a value of "-1" must be entered for the "address extension" parameter.

Example:

All user-specific auxiliary functions with the value "= 8" are assigned to the tenth auxiliary function group.

MD22000 $MC_AUXFU_ASSIGN_GROUP [ 1 ] = 10
MD22010 $MC_AUXFU_ASSIGN_TYPE [ 1 ] = "H"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [ 1 ] = -1
MD22030 $MC_AUXFU_ASSIGN_VALUE [ 1 ] = 8

Value

MD22030 $MC_AUXFU_ASSIGN_VALUE[<n>] (value of user-defined auxiliary functions)

The functionality of the "value" parameter is not defined in user-specific auxiliary functions. The value is generally used to activate the corresponding PLC user function.

Grouping together auxiliary functions

If all the auxiliary functions of the same type and address extension are assigned to the same auxiliary function group, a value of "-1" must be entered for the "value" parameter.

Example:
All user-specific auxiliary functions with the address extension "= 2" are assigned to the
eleventh auxiliary function group.

MD22000 $MC_AUXFU_ASSIGN_GROUP [ 2 ] = 11
MD22010 $MC_AUXFU_ASSIGN_TYPE [ 2 ] = "H"
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [ 2 ] = 2
MD22030 $MC_AUXFU_ASSIGN_VALUE [ 2 ] = -1

8.3.1.4 Output behavior

The "output behavior" of user-defined auxiliary functions can be parameterized via the machine
data:

MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>] (output behavior of user-defined auxiliary
functions)

For a description of the individual output parameters, see the "Output behavior (Page 401)"
section of the predefined auxiliary functions. The information given there can be applied
analogously to the output behavior of user-defined auxiliary functions.

8.4 Associated auxiliary functions

Function

Associated auxiliary functions are user-defined auxiliary functions that have the same effect
as the corresponding predefined auxiliary functions. User-defined auxiliary functions can be
associated for the following predefined auxiliary functions:

- M0 (stop)
- M1 (conditional stop)

Requirements

The precondition for association of a user-defined auxiliary function with one of the predefined
auxiliary functions mentioned is parameterization of a user-defined auxiliary function. Only "M"
is allowed as a "type" parameter of the user-defined auxiliary function.

Parameter assignment

Association of a user-defined auxiliary function with one of the predefined auxiliary functions
mentioned is set in the machine data:

MD22254 $MC_AUXFU_ASSOC_M0_VALUE (additional M function for program stop)
MD22256 $MC_AUXFU_ASSOC_M1_VALUE (additional M function for conditional stop)
Group assignment

The group assignment of an associated user-defined auxiliary function is always the group assignment of the corresponding predefined auxiliary function.

Application

Associated auxiliary functions can be used in:

- Main program
- Subroutine
- Cycle

Note

Associated auxiliary functions may not be used in synchronized actions.

NC/PLC interface signals

In the case of an associated user-defined auxiliary function, the same signals are output to the NC/PLC interface as for the corresponding predefined auxiliary function. To distinguish which auxiliary function has actually been programmed, the value of the user-defined auxiliary function ("value" parameter) is output as the value of the auxiliary function. This means it is possible to distinguish between predefined and user-defined auxiliary functions in the PLC user program.

Note

A change in machine data MD22254 and/or MD22256 may require corresponding adjustment of the PLC user program.

Specific NC/PLC interface signals

The following specific NC/PLC interface signals are available:

- DB21, ... DBX318.5 (associated M00/M01 active) feedback signal
- DB21, ... DBX30.5 (activate associated M01) activation signal

Boundary conditions

Please note the following boundary conditions:

- A user-defined auxiliary function may not be multiply associated.
- Predefined auxiliary functions (e.g. M3, M4, M5 etc.) may not be associated.

Example

Associating the user-defined auxiliary function M123 with M0:

MD22254 $MC_AUXFU_ASSOC_M0_VALUE = 123

The user-defined auxiliary function M123 thus has the same functionality as M0.
8.5 Type-specific output behavior

Function

The output behavior of auxiliary functions relative to a traversing motion programmed in the parts program block can be defined type-specifically.

Parameter assignment

Parameters are assigned to type-specific output behavior via the machine data:

- MD22200 $MC_AUXFU_M_SYNC_TYPE (output time for M functions)
- MD22210 $MC_AUXFU_S_SYNC_TYPE (output time for S functions)
- MD22220 $MC_AUXFU_T_SYNC_TYPE (output time for T functions)
- MD22230 $MC_AUXFU_H_SYNC_TYPE (output time for H functions)
- MD22240 $MC_AUXFU_F_SYNC_TYPE (output time for F functions)
- MD22250 $MC_AUXFU_D_SYNC_TYPE (output time for D functions)
- MD22252 $MC_AUXFU_DL_SYNC_TYPE (output time for DL functions)

The following output behaviors can be parameterized:

<table>
<thead>
<tr>
<th>Value</th>
<th>Output behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output prior to motion</td>
</tr>
<tr>
<td>1</td>
<td>Output during motion</td>
</tr>
<tr>
<td>2</td>
<td>Output at block end</td>
</tr>
<tr>
<td>3</td>
<td>No output to the PLC</td>
</tr>
<tr>
<td>4</td>
<td>Output according to the output behavior defined with MD22080</td>
</tr>
</tbody>
</table>

For a description of the various output behaviors, see the section titled "Output behavior (Page 401)".

Note

For the output behavior that can be set for each type of auxiliary function, please refer to the "Detailed Description of Machine Data" Parameter Manual.

Example

Output of auxiliary functions with different output behaviors in a part program block with traverse movement.
Output behavior for which parameters have been assigned:

MD22200 $MC_AUXFU_M_SYNC_TYPE = 1 ⇒ M function: Output during motion
MD22220 $MC_AUXFU_T_SYNC_TYPE = 0 ⇒ T function: Output prior to motion
MD22230 $MC_AUXFU_H_SYNC_TYPE = 2 ⇒ H function: Output at the end of the block

Parts program block:

```
... N10 G01 X100 M07 H5 T5 ...
...```

Time sequence for auxiliary function output:

8.6 Priorities of the output behavior for which parameters have been assigned

The following priorities must be observed for the following areas in connection with the parameterized output behavior of an auxiliary function:

- Output duration (normal / quick acknowledgement)
- Output relative to motion (prior to / during / after the motion)

As a general rule, the parameterized output behavior with lower priority becomes active if no output behavior with higher priority has been parameterized.
Output duration

The following priorities apply to the output duration:

<table>
<thead>
<tr>
<th>Priority</th>
<th>Output behavior</th>
<th>Defined via:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highest</td>
<td>Auxiliary function-specific</td>
<td>Part program instruction: QU(…) (see Section &quot;Programmable output duration (Page 414)&quot;)</td>
</tr>
</tbody>
</table>
| ↓        | Auxiliary function-specific | MD22035 $MC_AUXFU_ASSIGN_SYNC[<n>]
|          |                  | MD22080 $MC_AUXFU_PREDEF_SYNC[<n>] |
| ↓        | Group-specific   | MD11110 $MC_AUXFU_GROUP_SPEC[<n>] |
| Lowest   | Not defined      | Default output behavior: Output duration one OB1 cycle |

Output relative to motion

The following rules apply to output relative to motion:

<table>
<thead>
<tr>
<th>Priority</th>
<th>Output behavior</th>
<th>Defined via:</th>
</tr>
</thead>
</table>
| Highest  | Auxiliary function-specific | MD22035 $MC_AUXFU_ASSIGN_SYNC[<n>]
|          |                  | MD22080 $MC_AUXFU_PREDEF_SYNC[<n>] |
| ↓        | Group-specific   | MD11110 $MC_AUXFU_GROUP_SPEC[<n>] |
| ↓        | Type-specific    | MD22200 $MC_AUXFU_M_SYNC_TYPE
|          |                  | MD22210 $MC_AUXFU_S_SYNC_TYPE
|          |                  | MD22220 $MC_AUXFU_T_SYNC_TYPE
|          |                  | MD22230 $MC_AUXFU_H_SYNC_TYPE
|          |                  | MD22240 $MC_AUXFU_F_SYNC_TYPE
|          |                  | MD22250 $MC_AUXFU_D_SYNC_TYPE
|          |                  | MD22252 $MC_AUXFU_DL_SYNC_TYPE |
| Lowest   | Not defined      | Default output behavior: Output at block end |

Note

Part program blocks without path motion

In a part program block without a path motion (even those with positioning axes and spindles), the auxiliary functions are all output immediately in a block.

8.7 Programming an auxiliary function

Syntax

An auxiliary function is programmed in a part program block with the following syntax:

<Type>[<Address extension>=]<Value>
Note
If no address extension is programmed, the address extension is implicitly set = 0.
Predefined auxiliary functions with the address extension = 0 always refer to the master spindle
of the channel.

Symbolic addressing
The values for the "address extension" and "value" parameters can also be specified
symbolically. The symbolic name for the address extension must then be stated in brackets.
Example:
Symbolic programming of the auxiliary function M3 (spindle right) for the first spindle:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEF SPINDEL_NR=1 ; First spindle in the channel</td>
<td></td>
</tr>
<tr>
<td>DEF DREHRICHTUNG=3 ; Clockwise direction of rotation</td>
<td></td>
</tr>
<tr>
<td>N100 M[SPINDEL_NR] - DREHRICHTUNG ; corresponding to: M1=3</td>
<td></td>
</tr>
</tbody>
</table>

Note
If you use symbolic names to program an auxiliary function, the symbolic name is not
transferred when the auxiliary function is output to the PLC. The corresponding numerical value
is transferred instead.

Examples

Example 1: Programming of predefined auxiliary functions

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 M3 ; &quot;Spindle right&quot; for the master spindle of the channel.</td>
<td></td>
</tr>
<tr>
<td>N20 M0=3 ; &quot;Spindle right&quot; for the master spindle of the channel.</td>
<td></td>
</tr>
<tr>
<td>N30 M1=3 ; &quot;Spindle right&quot; for the 1st spindle of the channel.</td>
<td></td>
</tr>
<tr>
<td>N40 M2=3 ; &quot;Spindle right&quot; for the 2nd spindle of the channel.</td>
<td></td>
</tr>
</tbody>
</table>

Example 2: Programming examples of auxiliary functions with the corresponding values for
output to the PLC

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEF Coolant=12 ; Output to PLC: - - -</td>
<td></td>
</tr>
<tr>
<td>DEF Lubricant=130 ; Output to PLC: - - -</td>
<td></td>
</tr>
<tr>
<td>H[coolant]=lubricant ; Output to PLC: H12=130</td>
<td></td>
</tr>
<tr>
<td>H=coolant ; Output to PLC: H0=12</td>
<td></td>
</tr>
<tr>
<td>H5 ; Output to PLC: H0=5</td>
<td></td>
</tr>
<tr>
<td>H=5.379 ; Output to PLC: H0=5.379</td>
<td></td>
</tr>
</tbody>
</table>
### 8.8 Programmable output duration

**Function**

User-specific auxiliary functions, for which the output behavior "Output duration of an OB1 cycle (slow acknowledgement)" was parameterized, can be defined for individual outputs via the parts program guide QU (Quick) for auxiliary functions with quick acknowledgement.

**Syntax**

An auxiliary function with quick acknowledgment is defined in a part program block with the following syntax:

```
<Type>[<Address extension>]=QU(<Value>)
```

**Example**

Different behavior for the output of the auxiliary functions M100 and M200 in a parts program.

The output behavior of the auxiliary functions is parameterized as follows:

- **M100**
  - Output duration one OB1 cycle (slow acknowledgement)
  - Output during motion
- **M200**
  - Output duration one OB1 cycle (slow acknowledgement)
  - Output prior to motion

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>H17=3.5</td>
<td>Output to PLC: H17=3.5</td>
</tr>
<tr>
<td>H[coolant]=13.8</td>
<td>Output to PLC: H12=13.8</td>
</tr>
<tr>
<td>H='HFF13'</td>
<td>Output to PLC: H0=65299</td>
</tr>
<tr>
<td>H='B1110'</td>
<td>Output to PLC: H0=14</td>
</tr>
<tr>
<td>H5.3=21</td>
<td>Error</td>
</tr>
</tbody>
</table>

- **N10** G94 G01 X50 M100
  - Output of M100: during the motion
  - Acknowledgment: slow

- **N20** Y5 M100 M200
  - Output of M200: prior to the motion
  - Output of M100: during the motion
  - Acknowledgment: slow

- **N30** Y0 M=QU(100) M=QU(200)
  - Output of M200: prior to the motion
  - Output of M100: during the motion
  - Acknowledgement: quick

- **N40** X0
8.9 Auxiliary function output to the PLC

Function

On output of an auxiliary function to the PLC, the following signals and values are transferred to the NC/PLC interface:

- Change signals
- "Address extension" parameter
- "Value" parameter

Data areas in the NC/PLC interface

The change signals and values of the auxiliary functions are within the following data areas in the NC/PLC interface:

- Change signals for auxiliary function transfer from NC channel: DB21, ... DBB58 - DBB67
- Transferred M and S functions: DB21, ... DBB68 - DBB112
8.10 Auxiliary functions without block change delay

Function

For auxiliary functions with parameterized and/or programmed output behavior, too:

- "Output duration one OB40 cycle (quick acknowledgment)"
- "Output before the motion" or "Output during the motion"

there may be drops in velocity in continuos-path mode (short traverse paths and high velocities). This the system has to wait for acknowledgment of the auxiliary function by the PLC toward the end of the block. To avoid these velocity drops, the block change can be made irrespective of whether such auxiliary functions have been acknowledged:

Parameter assignment

Suppression of the block change delay with quick auxiliary functions is set via the machine data:

MD22100 $MC_AUXFU_QUICK_BLOCKCHANGE (block change delay with quick auxiliary functions)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>In the case of quick auxiliary function output to the PLC, the block change is delayed until acknowledgment by the PLC (OB40).</td>
</tr>
<tr>
<td>1</td>
<td>In the case of quick auxiliary function output to the PLC, the block change is not delayed.</td>
</tr>
</tbody>
</table>

Boundary conditions

Synchronism of auxiliary functions that are output without a block change delay is no longer ensured for the part program block in which they are programmed. In the worst case scenario, acknowledgment comes one OB40 cycle and execution of the auxiliary function comes one OB1 cycle after the change to the next part program block.
8.11 M function with an implicit preprocessing stop

Function

Triggering a preprocessing stop in conjunction with an auxiliary function can be programmed explicitly via the STOPRE part program command. Always triggering a preprocessing stop in M function programming can be parameterized for each M function via the following machine data:

MD10713 $MN_M_NO_FCT_STOPRE[<n>] (M function with preprocessing stop)

Example

The user-defined M function M88 is intended to trigger a preprocessing stop.

Parameter assignment:

MD10713 $MN_M_NO_FCT_STOPRE [0] = 88

Application:

Parts program (extract)

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 G0 X10</td>
<td>M88; Traversing motion and implicit preprocessing stop via M88</td>
</tr>
<tr>
<td>N110 Y=R1</td>
<td>N110 is only interpreted after the traversing motion has been completed and the M function has been acknowledged.</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Boundary conditions

If a subroutine called indirectly via an M function in a part program in one of the following ways, no preprocessing stop is performed:

- MD10715 $MN_M_NO_FCT_CYCLE (M function to be replaced by subroutine)
- M98 (ISO dialect T / ISO dialect M)

8.12 Response to overstore

Overstore

On the SINUMERIK operator interface, before starting the following functions:

- NC START of a part program
- NC START to resume an interrupted part program

the auxiliary functions that are output at the start can be changed by the "Overstore" function.
Possible applications include:
- Addition of auxiliary functions after block search
- Restoring the initial state to position a part program

Types of auxiliary functions that can be overstored
The following types of auxiliary functions can be overstored:
- M (special function)
- S (spindle speed)
- T (tool number)
- H (aux. function)
- D (tool offset number)
- DL (additive tool offset)
- F (feed)

Duration of validity
An overstored auxiliary function, e.g. M3 (spindle right), is valid until it is overwritten by another auxiliary function from the same auxiliary function group, by additional over storing or by programming in a part program block.

8.13 Behavior during block search

8.13.1 Auxiliary function output during type 1, 2, and 4 block searches

Output behavior
In the case of type 1, 2, and 4 block searches, the auxiliary functions are collected on the basis of specific groups. The last auxiliary function in each auxiliary function group is output after NC-START in a separate part program block before the actual reentry block, and has the following output behavior:
- Output duration of one OB1 cycle (normal acknowledgement)
- Output prior to motion
Output control

Whether or not the auxiliary function is output to the PLC after a block search can be configured via bit 8 of the machine data:

- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>]
  (output behavior of predefined auxiliary functions)
  where <n> = system function index (0 ... 32)

- MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>]
  (output behavior of user-defined auxiliary functions)
  where <n> = auxiliary function index (0 ... 254)

- MD11110 $MN_AUXFU_GROUP_SPEC[<n>]
  (output behavior of the auxiliary functions in a group)
  where <n> = group index (0 ... 63)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>Output during type 1, 2, and 4 block searches</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>No output during type 1, 2, and 4 block searches</td>
</tr>
</tbody>
</table>

This behavior does not affect the display and does not affect variables $AC_AUXFU_STATE[<n>], $AC_AUXFU_VALUE[<n>], and $AC_AUXFU_EXT[<n>]. The auxiliary functions are always regarded as collected after a block search, even though they are not output to the PLC.

During collection, an auxiliary function that is not output after a block search also overwrites an auxiliary function whose bit 8 is not set.

The user can scan the collected auxiliary functions after a block search and, under certain circumstances, output them again by means of the subprogram or synchronized actions.

Note

The following auxiliary functions are not collected:

- Auxiliary functions which are not assigned to any auxiliary function group.
- Auxiliary functions which are assigned to the first auxiliary function group.

Overstorage of auxiliary functions

After completion of a block search, the collected auxiliary functions are output on the next NC-START. If it is necessary to output additional auxiliary functions, they can be added via the "Overstore" function (see Section "Response to overstore (Page 417)").

M19 behavior (position spindle)

After a block search, the last spindle positioning command programmed with M19 is always carried out, even if other spindle-specific auxiliary functions are programmed between the part program with M19 and the target block. Setting the necessary spindle enables must therefore be derived from the interface signals of the traverse commands in the PLC user program:

DB31, ... DBX64.6/64.7 (traversing command minus/plus)
In this case, the spindle-specific auxiliary functions M3, M4, and M5 are not suitable because they might not be output to the PLC until after the spindle positioning.

For detailed information on the block search, see Section "K1: Mode group, channel, program operation, reset response" (Page 455).

### 8.13.2 Assignment of an auxiliary function to a number of groups

**Function**

User-defined auxiliary functions can also be assigned to multiple groups via the group assignment (MD22000 $MC_AUXFU_ASSIGN_GROUP). During the block search these auxiliary functions are collected for all the configured groups.

**Note**

Predefined auxiliary functions can only be assigned to one group.

**Example**

The DIN includes the following M-commands for coolant output:

- M7: Coolant 2 ON
- M8: Coolant 1 ON
- M9: Coolants 1 and 2 OFF

Consequently, both coolants can also be active together:

- If M7 and M8 are collected in two separate groups (e.g. groups 5 and 6)
- If M9 has to be assigned to these two groups, e.g.
  - Group 5: M7, M9
  - Group 6: M8, M9

**Parameterization:**

MD11100 $MN_AUXFU_MAXNUM_GROUP_ASSIGN = 4
MD22000 $MC_AUXFU_ASSIGN_GROUP [0] = 5
MD22000 $MC_AUXFU_ASSIGN_GROUP [1] = 5
MD22000 $MC_AUXFU_ASSIGN_GROUP [2] = 6
MD22000 $MC_AUXFU_ASSIGN_GROUP [3] = 6
MD22010 $MC_AUXFU_ASSIGN_TYPE [0] = M
MD22010 $MC_AUXFU_ASSIGN_TYPE [1] = M
MD22010 $MC_AUXFU_ASSIGN_TYPE [2] = M
MD22010 $MC_AUXFU_ASSIGN_TYPE [3] = M
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [0] = 0
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [1] = 0
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [2] = 0
MD22020 $MC_AUXFU_ASSIGN_EXTENSION [3] = 0
MD22030 $MC_AUXFU_ASSIGN_VALUE [0] = 7
MD22030 $MC_AUXFU_ASSIGN_VALUE [1] = 9
MD22030 $MC_AUXFU_ASSIGN_VALUE [2] = 8
MD22030 $MC_AUXFU_ASSIGN_VALUE [3] = 9
MD22035 $MC_AUXFU_ASSIGN_SPEC [0] = 'H121'
MD22035 $MC_AUXFU_ASSIGN_SPEC [1] = 'H121'
MD22035 $MC_AUXFU_ASSIGN_SPEC [2] = 'H121'
MD22035 $MC_AUXFU_ASSIGN_SPEC [3] = 'H121'

Part program (section):

<table>
<thead>
<tr>
<th>Program code</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>N10 ... M8</td>
</tr>
<tr>
<td>N20 ... M9</td>
</tr>
<tr>
<td>N30 ... M7</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

During the block search, the auxiliary function M9 is collected for groups 5 and 6.

Scan of the collected M auxiliary functions:

M function of the fifth group: $AC_AUXFU_M_VALUE [4] = 7
M function of the sixth group: $AC_AUXFU_M_VALUE [5] = 9

8.13.3 Time stamp of the active M auxiliary function

When outputting collected auxiliary functions following a block search, attention must be paid to the sequence during collecting. For this reason, each group is assigned a time stamp which can be queried on a group-specific basis by way of the system variable below:

$AC_AUXFU_M_TICK[<n>] (time stamp of the active M auxiliary function)

8.13.4 Determining the output sequence

Function

The following predefined procedure is provided to simplify the process of determining the output sequence of M auxiliary functions for the programmer:
AUXFUMSEQ(VAR INT _NUM_IN, VAR INT _M_IN[], VAR INT _EXT_IN[], VAR INT _NUM_OUT, VAR INT _M_OUT[], VAR INT _EXT_OUT[])

**Input parameters:**
- VAR INT _NUM_IN: Number of relevant M commands
- VAR INT _M_IN[]: Field of relevant M codes
- VAR INT _EXT_IN[]: Field of relevant M address extensions

**Output parameters:**
- VAR INT _NUM_OUT: Number of determined M codes
- VAR INT _M_OUT[]: Field of determined M codes
- VAR INT _EXT_OUT[]: Field of determined M address extensions

The function determines the sequence in which the M auxiliary functions, which have been collected on a group-specific basis, are output for the predefined M codes. The sequence is determined from the collection times $\text{AC_AUXFU}_M\_\text{TICK}[<n>]$ (see Section "Time stamp of the active M auxiliary function (Page 421)").

A particular M code is only taken into account once, even if it belongs to more than one group. If the number of relevant M commands is less than or equal to 0, all the collected M codes are output. The number of relevant M commands is limited to 64.

**Example**

M commands for coolant output:
- M7: Coolant 2 ON
- M8: Coolant 1 ON
- M9: Coolants 1 and 2 OFF

Group assignment:
- Group 5: M7, M9
- Group 6: M8, M9

Part program (section):

```
<table>
<thead>
<tr>
<th>Program code</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>N10 ... M8</td>
</tr>
<tr>
<td>N20 ... M9</td>
</tr>
<tr>
<td>N30 ... M7</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
```
During block searches, the auxiliary functions are collected on the basis of specific groups. The last auxiliary function in an auxiliary function group is output to the PLC following a block search:

- Group 5: M7
- Group 6: M9

If they are output in the sequence M7 → M9, no coolant is then active. However, coolant 2 would be active during the execution of the program. Therefore, the correct output sequence for the M auxiliary functions is determined with an ASUP which contains the predefined procedure `AUXFUMSEQ(...)`:

```plaintext
Program code
DEF INT _I, _M_IN[3], _EXT_IN[3], _NUM_OUT, _M_OUT[2], _EXT_OUT[2]
_M_IN[0]=7 _EXT_IN[0]=0
_M_IN[1]=8 _EXT_IN[1]=0
AUXFUMSEQ(3,_M_IN,_EXT_IN,_NUM_OUT,_M_OUT,_EXT_OUT)
FOR _I = 0 TO _NUM_OUT-1
  M[_EXT_OUT[_I]]=_M_OUT[_I]
ENDFOR
```

### 8.13.5 Output suppression of spindle-specific auxiliary functions

#### Function

In certain situations, such as a tool change, it may be necessary not to output the spindle-specific auxiliary functions collected during the block search in action blocks, but to delay output, for example, until after a tool change. The automatic output of the spindle-specific auxiliary functions after a block search may be suppressed for this purpose. Output can then be performed manually later by overstoring or by an ASUB.

#### Parameterization

Suppression of the automatic output of the spindle-specific auxiliary functions after a block search is set via machine data:

MD11450 $MN_SEARCH_RUN_MODE (behavior after a block search)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>The output of the spindle-specific auxiliary functions is performed in the action blocks.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Output of the auxiliary functions is suppressed in the action blocks.</td>
</tr>
</tbody>
</table>
System variables

The spindle-specific auxiliary functions are always stored in the following system variables during block searches, irrespective of the parameter assignment described above:

<table>
<thead>
<tr>
<th>System variable</th>
<th>Description</th>
<th>Range of values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PSEARCH_S [&lt;n&gt;]</td>
<td>Accumulated spindle speed</td>
<td>0 ... Smax</td>
</tr>
<tr>
<td>$PSEARCH_SDIR [&lt;n&gt;]</td>
<td>Accumulated spindle direction of rotation</td>
<td>3, 4, 5, -5, -19, 70</td>
</tr>
<tr>
<td>$PSEARCH_SGEAR [&lt;n&gt;]</td>
<td>Accumulated spindle gear stage M function</td>
<td>40 ... 45</td>
</tr>
<tr>
<td>$PSEARCH_SPOS [&lt;n&gt;]</td>
<td>Accumulated spindle position</td>
<td>0 ... MD30330 $MA_MODULO_RANGE (size of the module range) or Accumulated traversing path -100.000.000 ... 100.000.000</td>
</tr>
<tr>
<td>$PSEARCH_SPOSMODE [&lt;n&gt;]</td>
<td>Accumulated position approach mode</td>
<td>0 ... 5</td>
</tr>
</tbody>
</table>

For later output of the spindle-specific auxiliary functions, the system variables can be read in an ASUB, for example, and output after the action blocks are output:

```
DB21, ... DBX32.6 = 1 (last action block active)
```

**Note**

The contents of the system variables $P_S$, $P_DIR$ and $P_SGEAR$ may be lost after block search due to synchronization operations.

For more detailed information on ASUB, block search and action blocks, see Section "K1: Mode group, channel, program operation, reset response (Page 455)".

**Example**

Block search for contour with suppression of output of the spindle-specific auxiliary functions and start of an ASUB after output of action blocks.

Parameterization: MD11450 $MN_SEARCH_RUN_MODE, bit 2 = 1

After the block search on N55, the ASUB is started.

Part program:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N05 M3 S200</td>
<td>; Spindle 1</td>
</tr>
<tr>
<td>N10 G4 F3</td>
<td></td>
</tr>
<tr>
<td>N15 SPOS=111</td>
<td>; Spindle 1 is positioned to 111 degrees in the ASUB</td>
</tr>
<tr>
<td>N20 M2=4 S2=300</td>
<td>; Spindle 2</td>
</tr>
<tr>
<td>N25 G4 F3</td>
<td></td>
</tr>
</tbody>
</table>
**Explanation of example**

If the number of spindles is known, outputs of the same type can be written in one part program block to reduce program runtime.

Output of $P\_SEARCH\_SDIR$ should be made in a separate part program block because spindle positioning or switchover to axis mode in conjunction with the gear change can cause an alarm.

If the ASUB which has been started is ended with REPOSA, spindle 1 remains at position 111 degrees, while spindle 2 is repositioned at position 77 degrees.

If a different response is required, the program sequence for block search (for example) "N05 M3 S..." and "N30 SPOS[2] = IC(...)" requires special treatment.

Whether block search is active can be ascertained in the ASUB via the system variable $P\_SEARCH$.

$P\_SEARCH==1$ ; Block search active

In the case of incremental positioning after speed control operation, the path to be traversed is defined but, in some cases, the final position reached only becomes known during positioning. This is the case, for example, during position calibration on crossing the zero mark when switching on position control. For this reason, the distance programmed after the zero position is accepted as the REPOS position (REPOSA in the ASUB).

### Supplementary conditions

**Collected S values**
The meaning of an S value in the part program depends on the feed type that is currently active:

- G93, G94, G95, G97, G971: The S value is interpreted as the speed
- G96, G961: The S value is interpreted as a constant cutting rate

If the feed operation is changed (e.g. for a tool change) before output of the system variable \$P\_SEARCH\_S, the original setting from the target block in the part program must be restored to avoid use of the wrong type of feed.

**Collected direction of rotation**

For output of the direction of rotation, the system variable \$P\_SEARCH\_SDIR is assigned default value "-5" at the start of the block search. This value has no effect on output.

This ensures that the last spindle operating mode is retained for a block search across program sections in which spindles are not programmed with a direction of rotation, positioning or axis mode.

The programming of M19, SPOS, and SPOSA is collected as "M-19" (internal M19) in the system variables \$P\_SEARCH\_SDIR as an alternative to M3, M4, M5, and M70.

For the output of "M-19", the positioning data is read internally from the system variables \$P\_SEARCH\_SPOS and \$P\_SEARCH\_SPOSMODE. Both system variables can also be written to, in order, for example, to make corrections.

**Note**

Because of the assignments described above (e.g. M[<n>] = \$P\_SEARCH\_SDIR[<n>]), the values "-5" and "-19" generally remain hidden from the user and only have to be observed in the case of special evaluation of the system variables in the ASUB.

---

### 8.13.6 Auxiliary function output with a type 5 block search (SERUPRO)

**Output behavior**

In the case of type 5 block searches (SERUPRO), an auxiliary function can be output to the PLC during the block search and/or collected on a group-specific basis in the following system variables:

- \$AC\_AUXFU\_PREDEF\_INDEX[<n>] (index of a predefined auxiliary function)
- \$AC\_AUXFU\_TYPE[<n>] (type of auxiliary function)
- \$AC\_AUXFU\_STATE[<n>] (output state of the auxiliary function)
- \$AC\_AUXFU\_EXT[<n>] (address extension of the auxiliary function)
- \$AC\_AUXFU\_VALUE[<n>] (value of the auxiliary function)

For a description of the system variables, see Section "Querying system variables (Page 440)".
Output control

Whether an auxiliary function is output to the PLC during a type 5 block search (SERUPRO) and/or collected on a group-specific basis in the following system variables can be configured via bits 9 and 10 of the machine data:

- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>]
  (output behavior of predefined auxiliary functions)
  where <n> = system function index (0 ... 32)

- MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>]
  (output behavior of user-defined auxiliary functions)
  where <n> = auxiliary function index (0 ... 254)

- MD11110 $MN_AUXFU_GROUP_SPEC[<n>]
  (output behavior of the auxiliary functions in a group)
  where <n> = group index (0 ... 63)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0</td>
<td>No collection during type 5 block searches (SERUPRO)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Collection during type 5 block searches (SERUPRO)</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>Output during type 5 block searches (SERUPRO)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>No output during type 5 block searches (SERUPRO)</td>
</tr>
</tbody>
</table>

Output counter

The user can output the collected auxiliary functions to the PLC on a channel-by-channel basis in the block search ASUB. For the purposes of serialized output via multiple channels, the three output counters are changed across all the channels each time an auxiliary function is output:

<table>
<thead>
<tr>
<th>System variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_AUXFU_TICK[&lt;n&gt;,&lt;m&gt;]</td>
<td>Output counter of the active auxiliary function</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;n&gt;</td>
<td>Group index (0 to 63)</td>
</tr>
<tr>
<td>&lt;m&gt;</td>
<td>Output counter (0 ... 2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output sequence counter (all outputs within an IPO cycle)</td>
</tr>
<tr>
<td>1</td>
<td>Package counter within an output sequence in the IPO cycle</td>
</tr>
<tr>
<td>2</td>
<td>Auxiliary function counter within a package</td>
</tr>
</tbody>
</table>
Explanation

- An auxiliary function package comprises a maximum of ten auxiliary functions.
- Two packages can be processed per IPO cycle in each channel during SERUPRO because synchronized actions are processed in this cycle.
- An output sequence of up to a maximum of 20 packages (2 packages per channel * 10 channels) can be processed within an IPO cycle across all channels.

The encoding indicates how many auxiliary function packages and, within these, how many auxiliary functions have been processed during the same IPO cycle:

- Auxiliary functions which have been collected in one IPO cycle have the same sequence counter.
- Auxiliary functions which have been collected in one package (block or synchronized action) all have the same package counter.

The total on the auxiliary function counter increases every time an auxiliary function is collected.

Global list of auxiliary functions

At the end of SERUPRO, the auxiliary functions collected on a group-specific basis in the individual channels are entered in a cross-channel list with the channel number and group index according to their counter state.

<table>
<thead>
<tr>
<th>System variable 1)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_AUXFU_TICK[&lt;n&gt;,&lt;m&gt;]</td>
<td>Counter value</td>
</tr>
<tr>
<td>$AN_AUXFU_LIST_CHANNO[&lt;n&gt;] 2)</td>
<td>Channel number</td>
</tr>
<tr>
<td>$AN_AUXFU_LIST_GROUPINDEX[&lt;n&gt;] 2)</td>
<td>Group index</td>
</tr>
</tbody>
</table>

1) Value range index <n>: 0 ... MAXNUM_GROUPS * MAXNUM_CHANNELS - 1
2) The system variables can be read and written.
The global list is structured on the basis of the sequence in which the search target was found. It is intended to be used as a system proposal for auxiliary functions to be output in the following ASUB at the end of SERUPRO. If an auxiliary function is not to be output, the corresponding group index is to be set to "0".

**Behavior regarding spindle auxiliary functions**

Following the start of the search, all the channels collect the auxiliary functions in the channel variables on a group-specific basis. In order to perform a far-reaching restoration of the spindle state in the SERUPRO target block using the collected auxiliary functions, the last active auxiliary function in any group of spindle auxiliary functions must characterize the state of the spindle in the target block. In the case of transitions in spindle states, obsolete auxiliary functions are deleted or, if necessary, implicit auxiliary functions are entered.

All the spindle auxiliary functions from the global auxiliary function list must correspond to the spindle states achieved in the target block to enable the auxiliary functions to be processed when the list is output and to ensure that no alarms or unintended spindle states are requested which could prevent the continuation of the part program.
This affects the groups of auxiliary functions for any spindle configured in the system, whereby the spindle number corresponds to an auxiliary function's address extension.

Group a: M3, M4, M5, M19, M70  
Group b: M40, M41, M42, M43, M44, M45  
Group c: S

Deleting obsolete auxiliary functions
In the functions below, the auxiliary functions from group a are deleted for the spindle concerned:

- For the following spindle when a generic coupling, such as COUPON, TRAILON, EGON, etc. is switched on

Generating implicit auxiliary functions from group a
In the functions below, the auxiliary functions from group a are generated implicitly for the spindle concerned:

- For the following spindle when the synchronous spindle coupling is switched off
  - COUPOF generates M3, M4 and S or M5 in the main run depending on the coupling situation.
  - COUPOF(S<n>, S<m>, POS) and COUPOFS(S<n>, S<m>, POS, POS) generate M3, M4 and S.
  - COUPOFS generates M5 in the main run.
  - COUPOFS(S<n>, S<m>, POS) generates M19 in the main run. The implicit M19 ("SPOS[<address extension>] = IC(0)" in the ASUB) activates the positioning mode without a traversing motion.

- M70 is generated during a traversing motion as an axis or during the transition to axis mode by selecting a transformation during which the spindle enters as an axis.

- M5 is generated during SPCOF.

Note
Within the context of the "axis interchange" and "axis container rotation" functions, the auxiliary functions for programming the spindle must always be specified in a way which ensures compatibility with the actual (motor) state during interchange/rotation. A distinction is made here between the axis interchange and axis container mechanisms.

Example of axis container rotation:
An axis container has four spindles, each assigned to a separate channel (1 - 4). M3 S1000 is always programmed in channel 1, and an axis container rotation is then executed. The other channels do not perform any spindle programming. Following the third axis container rotation and the fourth spindle programming, M3, all four spindles rotate clockwise at a speed of 1,000 rpm. If the end of the SERUPRO now lies within this range, every ASUB for a channel is expected to contain an M3 S1000 for the local spindle.

During interchange however, the collected auxiliary functions may only be assigned to the channel where the spindle is currently located.
Cross-channel auxiliary function

An auxiliary function can also be collected on a cross-channel basis in the global auxiliary function list in the case of type 5 block searches (SERUPRO). Only the last auxiliary function collected from this group (highest counter state) is entered in the global list.

The appropriate configuring is performed with the following machine data:

- MD22080 $MC_AUXFU_PREDEF_SPEC[<n>], Bit 11 (output behavior of predefined auxiliary functions) where <n> = system function index
- MD22035 $MC_AUXFU_ASSIGN_SPEC[<n>], Bit 11 (output behavior of user-defined auxiliary functions) where <n> = auxiliary function index
- MD11110 $MN_AUXFU_GROUP_SPEC[<n>], Bit 11 (output behavior of the auxiliary functions in a group) where <n> = group index

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>0</td>
<td>Channel-specific collection</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Cross-channel collection</td>
</tr>
</tbody>
</table>

The spindle auxiliary functions are filtered out beforehand at the end of the search depending on the spindle state. The channel data is updated accordingly. The global auxiliary function list can be processed sequentially in the ASUBs at the end of the SERUPRO, and the sorted auxiliary functions can be output with channel synchronization.

Querying the last auxiliary function collected

The index of the last auxiliary function collected in the global list can be queried using the system variable $AN_AUXFU_LIST_ENDINDEX.

8.13.7 ASUB at the end of the SERUPRO

Function

After completing the block search with the program test (SERUPRO), before starting the subsequent processing, the auxiliary functions collected during the search must be output. For this purpose, during the block search, the auxiliary functions are collected in a global list. The SERUPRO end ASUBs generate the corresponding part program blocks channel-specific from this list. This ensures that the collected auxiliary functions can be output both channel-specific as well as cross-channel in the correct sequence. A fully functional SERUPRO end ASUB is a component of the NCK software.

Users/machinery construction OEMs can change the SERUPRO end ASUB. The subsequently described functions support processing the global list of auxiliary functions and generating the part program blocks required for synchronized auxiliary function output.
Function AUXFUSYNC(…)

Function:
The function AUXFUSYNC generates a complete part program block as string from the global list of auxiliary functions at each call. The part program block either contains auxiliary functions or commands to synchronize auxiliary function outputs (WAITM, G4, etc.). The function triggers a preprocessing stop.

Syntax:
PROC AUXFUSYNC(VAR INT <NUM>, VAR INT <GROUPINDEX>[10], VAR STRING[400] <ASSEMBLED>)

Parameters:

<NUM>:
Contains information about the part program block, supplied in parameter <ASSEMBLED> or the auxiliary functions contained in it.
Value range: -1, 0, 1 ... 10

Value Meaning
≥1 Number of auxiliary functions contained in the part program block
0 Part program block without auxiliary functions, e.g. WAITM, G4
-1 End identifier. The global list of auxiliary functions has been completely processed for the actual channel.

<GROUP INDEX>:
Contains the indices of the auxiliary function groups contained in the part program block. With index = number of the auxiliary function group - 1

<ASSEMBLED>:
Contains the complete part program block for the channel-specific SERURO end ASUB as string.

Further information:
If auxiliary functions were collected via a synchronized action, two NC blocks are generated. One NC block to output the auxiliary functions. An executable NC block via which the NC block is transported to the main run to output the auxiliary functions:
1. Output of the auxiliary functions via synchronized action, e.g.: WHEN TRUE DO M100 M102
2. Executable NC block, e.g.: G4 F0.001

Function AUXFUDEL(…)

Function:
The function AUXFUDEL deletes the specified auxiliary function from the global list of auxiliary functions channel-specific for the calling channel. Deletion is realized by setting the corresponding group index …GROUPINDEX[n] to 0.
The function must be called before calling AUXFUSYNC.
The function triggers a preprocessing stop.

Syntax:
PROC AUXFUDEL(CHAR <TYPE>, INT <EXTENSION>, REAL <VALUE>, INT <GROUP>)
Parameters:

- <TYPE>: Type of auxiliary function to be deleted
- <EXTENSION>: Address extension of the auxiliary function to be deleted
- <VALUE>: Value of auxiliary function to be deleted
- <GROUP>: Number of the auxiliary function group

Function AUXFUDELG(...)

Function:
The function AUXFUDELG deletes all auxiliary functions of the specified auxiliary function group from the global list of auxiliary functions channel-specific for the calling channel. Deletion is realized by setting the corresponding group index ...GROUPINDEX[n] to 0.

The function must be called before calling AUXFUSYNC.
The function triggers a preprocessing stop.

Syntax:
PROC AUXFUDELG(INT <GROUP>)

Parameters:

- <GROUP>: Number of the auxiliary function group

Multi-channel block search

<table>
<thead>
<tr>
<th>NOTICE</th>
</tr>
</thead>
</table>

Multi-channel block search and AUXFUDEL / AUXFUDELG

If, for a multi-channel block search in the SERUPRO end ASUBs, auxiliary functions with AUXFUDEL / AUXFUDELG are deleted from the global list of auxiliary functions, before calling the AUXFUSYNC function, the channels involved must be synchronized. The synchronization ensures that before calling the AUXFUSYNC, all delete requests are processed and a consistent list is available.

Examples

Two examples for configuring a user-specific SERUPRO end ASUB.

Example 1: Deleting auxiliary functions and generating the auxiliary function output with AUXFUSYNC(...)

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 DEF STRING[400] ASSEMBLED=&quot;&quot;</td>
<td></td>
</tr>
<tr>
<td>N20 DEF STRING[31] FILENAME=&quot;/_CST_DIR/_AUXFU_SPF&quot;</td>
<td></td>
</tr>
<tr>
<td>N30 DEF INT GROUPINDEX[10]</td>
<td></td>
</tr>
<tr>
<td>N40 DEF INT NUM</td>
<td></td>
</tr>
<tr>
<td>N60 DEF INT ERROR</td>
<td></td>
</tr>
</tbody>
</table>
## Program code

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N140</td>
<td>AUXFUDEL(&quot;M&quot;,2,3,5)</td>
<td>; M2=3 (5th auxiliary function group) delete</td>
</tr>
<tr>
<td>N150</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N170</td>
<td>AUXFUDELG(6)</td>
<td>; Delete the collected auxiliary function of the 6th group.</td>
</tr>
<tr>
<td>N180</td>
<td>IF ISFILE(FILENAME)</td>
<td></td>
</tr>
<tr>
<td>N190</td>
<td>DELETE(ERROR,FILENAME)</td>
<td>; Delete the FILENAME file</td>
</tr>
<tr>
<td>N210</td>
<td>IF (ERROR&lt;&gt;0)</td>
<td>; Error evaluation</td>
</tr>
<tr>
<td>N220</td>
<td>SETAL(61000+ERROR)</td>
<td></td>
</tr>
<tr>
<td>N230</td>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>N240</td>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>N250</td>
<td>ENDIF</td>
<td>; CAUTION!</td>
</tr>
<tr>
<td>N270</td>
<td>LOOP</td>
<td></td>
</tr>
<tr>
<td>N300</td>
<td>AUXFUSYNC(NUM,GROUPINDEX,ASSEMBLED)</td>
<td>; General a part program block</td>
</tr>
<tr>
<td>N310</td>
<td>IF (NUM==-1)</td>
<td>; All auxiliary functions of the channel have been executed.</td>
</tr>
<tr>
<td>N340</td>
<td>GOTO LABEL1</td>
<td></td>
</tr>
<tr>
<td>N350</td>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>N380</td>
<td>WRITE(ERROR,FILENAME,ASSEMBLED)</td>
<td>; Write a part program block to file FILENAME.</td>
</tr>
<tr>
<td>N390</td>
<td>IF (ERROR&lt;&gt;0)</td>
<td>; Error evaluation</td>
</tr>
<tr>
<td>N400</td>
<td>SETAL(61000+ERROR)</td>
<td></td>
</tr>
<tr>
<td>N410</td>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>N430</td>
<td>ENDDO</td>
<td></td>
</tr>
<tr>
<td>N450</td>
<td>LABEL1:</td>
<td></td>
</tr>
<tr>
<td>N460</td>
<td>CALL FILENAME</td>
<td>; Process a generated subroutine.</td>
</tr>
<tr>
<td>N490</td>
<td>DELETE(ERROR,FILENAME)</td>
<td>; Delete the file again after execution.</td>
</tr>
<tr>
<td>N510</td>
<td>IF (ERROR&lt;&gt;0)</td>
<td></td>
</tr>
<tr>
<td>N530</td>
<td>SETAL(61000+ERROR)</td>
<td></td>
</tr>
<tr>
<td>N540</td>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>N550</td>
<td>M17</td>
<td></td>
</tr>
<tr>
<td>N560</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

### 8.13 Behavior during block search

- **Auxiliary function outputs to PLC**
- **H2: Auxiliary function outputs to PLC**
- **Function Manual, 01/2015, 6FC5397-0BP40-5BA2**
Example 2: Deleting auxiliary functions and generating the auxiliary function output without AUXFUSYNC(...)

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N0610 DEF STRING[400] ASSEMBLED=&quot;&quot;</td>
<td></td>
</tr>
<tr>
<td>N0620 DEF STRING[31] FILENAME=&quot;/N_CST_DIR/_N_AUXFU_SPF&quot;</td>
<td></td>
</tr>
<tr>
<td>N0630 DEF INT GROUPINDEX[10]</td>
<td></td>
</tr>
<tr>
<td>N0640 DEF INT NUM</td>
<td></td>
</tr>
<tr>
<td>N0650 DEF INT LAUF</td>
<td></td>
</tr>
<tr>
<td>N0660 DEF INT ERROR</td>
<td></td>
</tr>
<tr>
<td>N0670 DEF BOOL ISQUICK</td>
<td></td>
</tr>
<tr>
<td>N0680 DEF BOOL ISSYNACT</td>
<td></td>
</tr>
<tr>
<td>N0690 DEF BOOL ISIMPL</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N0760 AUXFUDEL(&quot;M&quot;,2,3,5)</td>
<td>; M2=3 (5th auxiliary function group) delete</td>
</tr>
<tr>
<td>N0770</td>
<td></td>
</tr>
<tr>
<td>N0790 AUXFUDELG(6)</td>
<td>; Delete the collected auxiliary function of the</td>
</tr>
<tr>
<td>N0800</td>
<td>; 6. group.</td>
</tr>
<tr>
<td>N0810 IF ISFILE(FILENAME)</td>
<td></td>
</tr>
<tr>
<td>N0830 DELETE(ERROR,FILENAME)</td>
<td>; File already exists and must be</td>
</tr>
<tr>
<td>N0840 IF (ERROR&lt;&gt;0)</td>
<td>; deleted.</td>
</tr>
<tr>
<td>N0850 SETAL(61000+ERROR)</td>
<td></td>
</tr>
<tr>
<td>N0860 ENDIF</td>
<td></td>
</tr>
<tr>
<td>N0870 ENDIF</td>
<td></td>
</tr>
<tr>
<td>N0880</td>
<td>; CAUTION!</td>
</tr>
<tr>
<td>; If, for a multi-channel block search, auxiliary functions with AUXFUDEL/AUXFUDELG</td>
<td></td>
</tr>
<tr>
<td>; are deleted from the global list of auxiliary functions, before the loop to</td>
<td></td>
</tr>
<tr>
<td>; generate the subprogram FILENAME with AUXFUSYNC, the channels must be synchronized.</td>
<td></td>
</tr>
<tr>
<td>; The synchronization ensures that all delete requests were processed</td>
<td></td>
</tr>
<tr>
<td>; in all channels and a consistent list is available.</td>
<td></td>
</tr>
<tr>
<td>; Example: WAITM(99,1,2,3)</td>
<td></td>
</tr>
<tr>
<td>N0890 LOOP</td>
<td></td>
</tr>
<tr>
<td>N0920 AUXFUSYNC(NUM,GROUPINDEX,ASSEMBLED)</td>
<td>; Procedure to generate</td>
</tr>
<tr>
<td></td>
<td>; auxiliary function blocks from the global</td>
</tr>
<tr>
<td></td>
<td>; auxiliary function list.</td>
</tr>
<tr>
<td>N0930</td>
<td></td>
</tr>
<tr>
<td>N0940 IF (NUM==1)</td>
<td>; All auxiliary functions of the channel</td>
</tr>
<tr>
<td>N0960 GOTO LABEL1</td>
<td>; are processed.</td>
</tr>
<tr>
<td>N0970 ENDIF</td>
<td></td>
</tr>
<tr>
<td>N0980</td>
<td></td>
</tr>
<tr>
<td>N1000 IF (NUM&gt;0)</td>
<td>; If auxiliary functions are output,</td>
</tr>
<tr>
<td></td>
<td>; the block is generated.</td>
</tr>
<tr>
<td>Program code</td>
<td>Comment</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>N1010</td>
<td>ASSEMBLED=&quot;&quot;</td>
</tr>
<tr>
<td>N1020</td>
<td>FOR LAUF=0 TO NUM-1 ; Collected auxiliary functions for a block.</td>
</tr>
<tr>
<td>N1050</td>
<td>IF GROUPINDEX[LAUF]&lt;&gt;0 ; Auxiliary functions deleted from the global list have the group index 0.</td>
</tr>
<tr>
<td>N1090</td>
<td>ISQUICK=$AC_AUXFU_SPEC[GROUPINDEX[LAUF]] BAND'H2'</td>
</tr>
<tr>
<td>N1100</td>
<td>ISSYNACT=$AC_AUXFU_SPEC[GROUPINDEX[LAUF]] BAND'H1000'</td>
</tr>
<tr>
<td>N1130</td>
<td>ISIMPL=$AC_AUXFU_SPEC[GROUPINDEX[LAUF]] BAND'H2000'</td>
</tr>
<tr>
<td>N1180</td>
<td>IF ISSYNACT ; Assemble a block for the M auxiliary function output</td>
</tr>
<tr>
<td>N1190</td>
<td>ASSEMBLED= ASSEMBLED &lt;&lt; &quot;WHEN TRUE DO &quot;</td>
</tr>
<tr>
<td>N1210</td>
<td>ENDIF</td>
</tr>
<tr>
<td>N1230</td>
<td>IF (ISIMPL AND ($AC_AUXFU_VALUE[GROUPINDEX[LAUF]]==19))</td>
</tr>
<tr>
<td>N1240</td>
<td>ASSEMBLED= ASSEMBLED &lt;&lt; &quot;$AC_AUXFU_EXT[GROUPINDEX[LAUF]]&quot;=IC(0)&quot;</td>
</tr>
<tr>
<td>N1270</td>
<td>ELSE</td>
</tr>
<tr>
<td>N1280</td>
<td>ASSEMBLED= ASSEMBLED &lt;&lt; &quot;M(&quot; &lt;&lt; $AC_AUXFU_EXT[GROUPINDEX[LAUF]] &lt;&lt; &quot;)=IC(0)&quot;</td>
</tr>
<tr>
<td>N1300</td>
<td>ENDIF</td>
</tr>
<tr>
<td>N1320</td>
<td>ASSEMBLED= ASSEMBLED &lt;&lt; &quot;$AC_AUXFU_VALUE[GROUPINDEX[LAUF]]&quot;</td>
</tr>
<tr>
<td>N1350</td>
<td>IF ISQUICK</td>
</tr>
<tr>
<td>N1360</td>
<td>ASSEMBLED= ASSEMBLED &lt;&lt; &quot;$AC_AUXFU_VALUE[GROUPINDEX[LAUF]]&quot;</td>
</tr>
<tr>
<td>N1370</td>
<td>ENDIF</td>
</tr>
<tr>
<td>N1380</td>
<td>ENDIF</td>
</tr>
<tr>
<td>N1400</td>
<td>ENDIF</td>
</tr>
<tr>
<td>N1420</td>
<td>ENDFOR</td>
</tr>
<tr>
<td>N1450</td>
<td>WRITE(ERROR,FILENAME,ASSEMBLED) ; Write an auxiliary function block to a file.</td>
</tr>
<tr>
<td>N1470</td>
<td>IF ISSYNACT</td>
</tr>
<tr>
<td>N1480</td>
<td>ASSEMBLED=&quot;G4 F0.001&quot;</td>
</tr>
<tr>
<td>N1490</td>
<td>WRITE(ERROR,FILENAME,ASSEMBLED)</td>
</tr>
</tbody>
</table>
Implicitly output auxiliary functions

Implicitly output auxiliary functions are auxiliary functions which have not been programmed explicitly and which are also output by other system functions (e.g. transformation selection, tool selection, etc.). These implicit auxiliary functions do not lead to any system function; instead, the M codes are collected according to output behavior parameters assigned to them and/or are output to the PLC.
Parameterization

The M codes for auxiliary functions to be output implicitly are defined with the machine data:

- **MD22530 $MC_TOCARR_CHANGE_M_CODE (M code at toolholder change)**
  This machine data value indicates the number of the M code which is output when a toolholder is activated at the NC/PLC interface.
  If the value is positive, the unchanged M code is always output.
  If the value is negative, the number of the toolholder is added to the machine data value, and this number is output.

- **MD22532 $MC_GEOAX_CHANGE_M_CODE (M code when switching the geometry axes)**
  Number of the M code which is output when the geometry axes on the NC/PLC interface are switched.

- **MD22534 $MC_TRAFO_CHANGE_M_CODE (M code in the case of transformation changes)**
  Number of the M code which is output during a transformation switch of the geometry axes at the NC/PLC interface.

**Note**

No M code is output if the number of the M code being output or the MD22530/MD22532/MD22534 value is between 0 and 6, or is either 17 or 30. Whether or not an M code which is generated in this manner leads to conflicts with other functions is not monitored.

Output behavior

In the case of implicitly output auxiliary functions, bit 13 is set in machine data MD22080 or MD22035 (output behavior of predefined or user-defined auxiliary functions).

This bit can be queried via the system variable $AC_AUXFU_SPEC[<n>].

Implicitly output auxiliary function M19

To achieve uniformity in terms of how M19 and SPOS or SPOSA behave at the NC/PLC interface, auxiliary function M19 can be output to the NC/PLC interface in the event of SPOS and SPOSA (see Section "General functionality (Page 1205)").

The implicitly output auxiliary function M19 is collected during the block search.

8.15 Information options

Information about auxiliary functions (e.g. about the output status) is possible via:

- The group-specific modal M auxiliary function display on the user interface
- Querying system variables in part programs and synchronized actions
8.15.1 Group-specific modal M auxiliary function display

Function

The output status and acknowledgement status of M auxiliary functions can be displayed on the user interface on a group-specific basis.

Requirements

To implement function-oriented acknowledgement and display of M auxiliary functions, the auxiliary functions must be managed in the PLC and, thus, in the user program itself. Therefore, it is up to the PLC programmer to program the acknowledgement of these auxiliary functions. He has to know which auxiliary functions in which group have to be acknowledged.

Standard

M auxiliary functions that are not managed by means of the PLC are identified by the NC as "transferred" and output to the PLC. There is no functional acknowledgement for these auxiliary functions. All M-auxiliary functions collected after a block search are also displayed so that the operator knows which auxiliary functions will be output after a start following a block search.

PLC activities

In the case of auxiliary function groups that are managed by the PLC itself, the PLC user program must acknowledge all auxiliary functions of this groups when Apply and Function End are activated. The PLC programmer must know all the auxiliary functions of these groups.

Miscellaneous

Only the group-specific M auxiliary functions are displayed. The block-by-block display is also retained. Up to 15 groups can be displayed, whereby only the last M function of a group that was either collected or output to the PLC is displayed for each group. The M functions are presented in various display modes depending on their status:

<table>
<thead>
<tr>
<th>Status</th>
<th>Display mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auxiliary function is collected</td>
<td>Inverted with yellow font</td>
</tr>
<tr>
<td>Auxiliary function is output from NCK to PLC</td>
<td>Inverted</td>
</tr>
<tr>
<td>Auxiliary function has been transferred from NCK to PLC and transport acknowledgement has taken place</td>
<td>Black font on gray background</td>
</tr>
<tr>
<td>Auxiliary function is managed by the PLC and has been directly applied by the PLC.</td>
<td>Black font on gray background</td>
</tr>
<tr>
<td>Auxiliary function is managed by the PLC, and the function acknowledgement has taken place.</td>
<td>Black font on gray background</td>
</tr>
</tbody>
</table>
Display update

The display is organized in such a way that the collected auxiliary functions are always displayed first, before those that were managed by the PLC and before those that were managed by the NC. A collected auxiliary function is marked as collected until it has been output from the NCK to the PLC. PLC-managed auxiliary functions are retained until they are displaced by another auxiliary function. In the case of a reset, only the collected auxiliary functions and the NC-managed auxiliary functions are deleted.

8.15.2 Querying system variables

Function

Auxiliary functions can be queried on a group-specific basis via system variables in the part program and via synchronized actions:

$AC_AUXFU_... [<n>] = <value>

<table>
<thead>
<tr>
<th>system variables</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_AUXFU_PREDEF_INDEX[&lt;n&gt;]</td>
<td>&lt;value&gt;: Index of the last auxiliary function collected for an auxiliary function group (search) or</td>
</tr>
<tr>
<td></td>
<td>the last predefined auxiliary function to be output</td>
</tr>
<tr>
<td></td>
<td>Type : INT</td>
</tr>
<tr>
<td></td>
<td>If no auxiliary function has been output yet for the specified group or if the auxiliary function</td>
</tr>
<tr>
<td></td>
<td>is a user-defined auxiliary function, the variable supplies the value &quot;.-1&quot;.</td>
</tr>
<tr>
<td></td>
<td>&lt;n&gt;: Group index (0 to 63)</td>
</tr>
<tr>
<td></td>
<td>Note: A predefined auxiliary function can be uniquely identified via this variable.</td>
</tr>
<tr>
<td>$AC_AUXFU_TYPE[&lt;n&gt;]</td>
<td>&lt;value&gt;: Type of the last auxiliary function collected for an auxiliary function group (search) or</td>
</tr>
<tr>
<td></td>
<td>the last auxiliary function to be output</td>
</tr>
<tr>
<td></td>
<td>Type : CHAR</td>
</tr>
<tr>
<td></td>
<td>&lt;n&gt;: Group index (0 to 63)</td>
</tr>
<tr>
<td>$AC_AUXFU_EXT[&lt;n&gt;] or M function specific:</td>
<td>&lt;value&gt;: Address extension of the last auxiliary function</td>
</tr>
<tr>
<td>$AC_AUXFU_M_EXT[&lt;n&gt;]</td>
<td>collected for an auxiliary function group (search) or the last auxiliary function to be output</td>
</tr>
<tr>
<td></td>
<td>Type : INT</td>
</tr>
<tr>
<td></td>
<td>&lt;n&gt;: Group index (0 to 63)</td>
</tr>
</tbody>
</table>
### System Variables

<table>
<thead>
<tr>
<th>Description</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_AUXFU_VALUE[n]$</td>
<td>Value of the last auxiliary function collected for an auxiliary function group (search) or the last auxiliary function to be output</td>
</tr>
<tr>
<td>Type: REAL</td>
<td></td>
</tr>
<tr>
<td>&lt;n&gt;: Group index (0 to 63)</td>
<td></td>
</tr>
<tr>
<td>$AC_AUXFU_M_VALUE[n]$</td>
<td>Value: M-auxiliary function output to the PLC</td>
</tr>
<tr>
<td>Type: REAL</td>
<td></td>
</tr>
<tr>
<td>&lt;n&gt;: Group index (0 to 63)</td>
<td></td>
</tr>
<tr>
<td>$AC_AUXFU_SPEC[n]$</td>
<td>Value: Bit-encoded output behavior according to MD22080/MD22035 (or QU programming) of the last auxiliary function collected for an auxiliary function group (search) or the last auxiliary function to be output</td>
</tr>
<tr>
<td>Type: INT</td>
<td></td>
</tr>
<tr>
<td>&lt;n&gt;: Group index (0 to 63)</td>
<td></td>
</tr>
<tr>
<td>Note: This variable can be used to determine whether the auxiliary function should be output with a fast acknowledgement.</td>
<td></td>
</tr>
<tr>
<td>$AC_AUXFU_STATE[n]$</td>
<td>Output state of the last auxiliary function collected for an auxiliary function group (search) or the last auxiliary function to be output</td>
</tr>
<tr>
<td>Type: INT</td>
<td></td>
</tr>
<tr>
<td>Value range: 0 ... 5</td>
<td></td>
</tr>
<tr>
<td>0: No auxiliary function</td>
<td></td>
</tr>
<tr>
<td>1: M-auxiliary function was collected via a search</td>
<td></td>
</tr>
<tr>
<td>2: M-auxiliary function has been output to the PLC</td>
<td></td>
</tr>
<tr>
<td>3: M-auxiliary function has been output to the PLC and the transport acknowledgement has taken place</td>
<td></td>
</tr>
<tr>
<td>4: M-auxiliary function is managed by the PLC and has been applied by the PLC</td>
<td></td>
</tr>
<tr>
<td>5: M-auxiliary function is managed by the PLC, and the function acknowledgement has taken place</td>
<td></td>
</tr>
<tr>
<td>&lt;n&gt;: Group index (0 to 63)</td>
<td></td>
</tr>
</tbody>
</table>

### Example

All M-auxiliary functions of the 1st group will be stored in the order they are output

```plaintext
id=1 every $AC_AUXFU_M_STATE[0]==2 do $AC_FIFO[0,0]=
$AC_AUXFU_M_VALUE[0]
```

### References

For more information on the system variables, refer to:

List Manual, system variables
8.16 Supplementary conditions

8.16.1 General constraints

Spindle replacement
Because the auxiliary functions are parameterized channel-specifically, if function: "spindle replacement" is used, the spindle-specific auxiliary function must be parameterized immediately in all channels that use the spindles.

Tool management
If tool management is active, the following constraints apply:

- T and M<k> functions are not output to the PLC.
  
  **Note**
  
  k is the parameterized value of the auxiliary function for the tool change (default: 6):
  
  MD22560 $MC_TOOL_CHANGE_M_CODE (auxiliary function for tool change)

- If no address extension is programmed, the auxiliary function refers to the master spindle or the master tool holder of the channel.
  
  Definition of the master spindle:
  
  - MD20090 $MC_SPIND_DEF_MASTER_SPIND
  
  - Part program instruction: SETMS

  Definition of the master tool holder
  
  - MD20124 $MC_TOOL_MANAGEMENT_TOOLHOLDER
  
  - Part program instruction: SETMTH

Maximum number of auxiliary functions per part program block
A maximum of 10 auxiliary functions may be programmed in one part program block.

DL (additive tool offset)
The following restrictions apply to the DL function:

- Only one DL function can be programmed per part program block.

- If DL functions are used in synchronous actions, parameter: "Value" is not output to the PLC.
8.16.2 Output behavior

Thread cutting

During active thread cutting G33, G34 and G35, the following output behavior is always active for the spindle-specific auxiliary functions:

- M3 (spindle right)
- M4 (spindle left)

- Output duration of one OB40 cycle (quick acknowledgement)
- Output during motion

The spindle-specific auxiliary function M5 (spindle stop) is always output at the end of the block. The part program block that contains M5 is always ended with exact stop, i.e. even during active continuous-path mode.

Synchronized actions

With output auxiliary functions from synchronized actions, the parameterized output behavior is ignored except for the following parameters:

- Bit0: Output duration of one OB1 cycle (normal acknowledgement)
- Bit1: Output duration of one OB40 cycle (quick acknowledgement)

Auxiliary functions: M17 or M2/M30 (end of subprogram)

In its own part program block

If one of the auxiliary functions M17, M2 or M30 is programmed as the only auxiliary function in a part program block and an axis is still in motion, the auxiliary function is not output to the PLC until after the axis has stopped.

Overriding the parameterized output behavior

The parameterized output behavior of the auxiliary functions M17 or M2/M30 is overridden by the output behavior that is determined in the following machine data:

MD20800 $MC_SPF_END_TO_VDI, bit 0 (subprogram end / stop to PLC)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>The auxiliary functions M17 or M2/M30 (subprogram end) are not output to the PLC. Continuous-path mode is not interrupted at the end of the subprogram</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>The auxiliary functions M17 or M2/M30 (subprogram end) are output to the PLC.</td>
</tr>
</tbody>
</table>

Auxiliary function: M1 (conditional stop)

Overriding the parameterized output behavior

The parameterized output behavior of the auxiliary function M1 is overridden by the output behavior defined in the following machine data:
MD20800 $MC_SPF_END_TO_VDI, bit 1 (subprogram end / stop to PLC)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>The auxiliary function M01 (conditional stop) is always output to the PLC. A quick acknowledgement is ineffective, because M01 is permanently assigned to the first auxiliary function group and is therefore always output at the end of the block.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>The auxiliary function M01 (conditional stop) is only output to the PLC, if the function: &quot;Programmed stop&quot; is activated via the HMI user interface. In the case of a quick acknowledgement, the M1 is output to the PLC during the motion. While the function is not active, this does not interrupt continuous-path mode.</td>
</tr>
</tbody>
</table>

Part program blocks without traversing motion

In a part program block without a traversing motion, all auxiliary functions are output in a block immediately, irrespective of their parameterized output behavior.

Spindle-specific auxiliary function output only as information for the PLC user program

In certain controller situations, e.g. at the end of a block search, the collected spindle-specific auxiliary functions (e.g. M3, M4, M5, M19, M40...M45, M70) is output to the NC/PLC interface only for information purposes for the PLC user program. The controller generates a part program block (action block) in which the collected auxiliary functions are entered with a negative address extension. The corresponding system functions are then not executed.

Example: M(-2) = 41 request gear stage change for the 2nd spindle

8.17 Examples

8.17.1 Extension of predefined auxiliary functions

Task

Parameter assignment of auxiliary functions M3, M4, and M5 for the second spindle of the channel

Parameter assignment: M3

Requirements:

- Machine data index: 0 (first user-defined auxiliary function)
- auxiliary function group: 5
- Type and value: M3 (spindle right)
Address extension: 2 as appropriate for the 2nd spindle of the channel

Output behavior:
- Output duration one OB1 cycle (normal acknowledgment)
- Output prior to motion

Parameter assignment:

- MD22000 $MC_AUXFU_ASSIGN_GROUP[0] = 5
- MD22010 $MC_AUXFU_ASSIGN_TYPE[0] = "M"
- MD22020 $MC_AUXFU_ASSIGN_EXTENSION[0] = 2
- MD22030 $MC_AUXFU_ASSIGN_VALUE[0] = 3
- MD22035 $MC_AUXFU_ASSIGN_SPEC[0] = 'H21'

Parameter assignment: M4

Requirements:
- Machine data index: 1 (second user-defined auxiliary function)
- auxiliary function group: 5
- Type and value: M4 (spindle left)
- Address extension: 2 as appropriate for the 2nd spindle of the channel
- Output behavior:
  - Output duration one OB1 cycle (normal acknowledgment)
  - Spindle response following acknowledgment
  - Output during motion

Parameter assignment:

- MD22000 $MC_AUXFU_ASSIGN_GROUP[1] = 5
- MD22010 $MC_AUXFU_ASSIGN_TYPE[1] = "M"
- MD22020 $MC_AUXFU_ASSIGN_EXTENSION[1] = 2
- MD22030 $MC_AUXFU_ASSIGN_VALUE[1] = 4
- MD22035 $MC_AUXFU_ASSIGN_SPEC[1] = 'H51'

Parameter assignment: M5

Requirements:
- Machine data index: 2 (third user-defined auxiliary function)
- auxiliary function group: 5
- Type and value: M5 (spindle stop)
Address extension: 2 as appropriate for the 2nd spindle of the channel

Output behavior:
- Output duration one OB1 cycle (normal acknowledgment)
- Spindle response following acknowledgment
- Output at block end

Parameter assignment:

- MD22000 $MC_AUXFU_ASSIGN_GROUP [2] = 5
- MD22010 $MC_AUXFU_ASSIGN_TYPE [2] = "M"
- MD22020 $MC_AUXFU_ASSIGN_EXTENSION [2] = 2
- MD22030 $MC_AUXFU_ASSIGN_VALUE [2] = 5
- MD22035 $MC_AUXFU_ASSIGN_SPEC [2] = 'H91'

8.17.2 Defining auxiliary functions

Task

Parameter assignment of the auxiliary function-specific machine data for a machine with the following configuration:

Spindles
- Spindle 1: Master spindle
- Spindle 2: Second spindle

Gear stages
- Spindle 1: 5 gear stages
- Spindle 2: No gear stages

Switching functions for cooling water on/off
- Spindle 1
  - "ON" = M50
  - "OFF" = M51
- Spindle 2
  - "ON" = M52
  - "OFF" = M53
Requirements

Spindle 1 (master spindle)

Note

Default assignments

- The auxiliary functions M3, M4, M5, M70 and M1=3, M1=4, M1=5, M1=70 of spindle 1 (master spindle) are assigned as standard to the second auxiliary function group.
- All S and S1 values of spindle 1 (master spindle) are assigned to the third auxiliary function group by default.

- The gear stage last programmed is to be output after block search. The following auxiliary functions are assigned to the ninth auxiliary function group for this reason:
  - M40, M41, M42, M43, M44, M45
  - M1=40, M1=41, M1=42, M1=43, M1=44, M1=45
- The auxiliary functions M3, M4, M5, M70 and M1=3, M1=4, M1=5, M1=70 (second auxiliary function group) and S and S1 values (third auxiliary function group) should possess the following output behavior:
  - Output duration one OB40 cycle (quick acknowledgment)
  - Output prior to motion
- The auxiliary functions for gear changeover M40 to M45 and M1=40 to M1=45 (ninth auxiliary function group) should have the following output behavior:
  - Output duration one OB1 cycle (normal acknowledgment)
  - Output prior to motion

Spindle 2

- Only one M function for directional reversal may be programmed in one block. The direction of rotation last programmed is to be output after block search. The following auxiliary functions are assigned to the tenth auxiliary function group for this reason:
  - M2=3, M2=4, M2=5, M2=70
- All S2 values are assigned to auxiliary function group 11.
- The auxiliary functions M2=3, M2=4, M2=5, M2=70 (tenth auxiliary function group) and S2 values (auxiliary function group 11) should have the following output behavior:
  - Output duration one OB40 cycle (quick acknowledgment)
  - Output prior to motion
Cooling water

- It is not permissible to switch the cooling water on and off in one part program block. After a block search, the cooling water will be switched on or off. For this purpose, the following auxiliary functions are assigned, for example, to auxiliary function group 12 or 13:
  - Auxiliary function group 12: M50, M51
  - Auxiliary function group 13: M52, M53

- The auxiliary functions M50, M51 (auxiliary function group 12) and M52, M53 (auxiliary function group 13) should have the following output behavior:
  - Output duration one OB1 cycle (normal acknowledgment)
  - Output prior to motion

Parameterization of the machine data

The machine data are parameterized by appropriate programming within a parts program.

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MN_AUXFU_MAXNUM_GROUP_ASSIGN=21</td>
<td>; Number of user-defined auxiliary functions per channel</td>
</tr>
<tr>
<td>$MN_AUXFU_GROUP_SPEC[1]='H22'</td>
<td>; Output behavior of auxiliary function group 2</td>
</tr>
<tr>
<td>$MN_AUXFU_GROUP_SPEC[2]='H22'</td>
<td>; Output behavior of auxiliary function group 3</td>
</tr>
<tr>
<td>$MN_AUXFU_GROUP_SPEC[8]='H21'</td>
<td>; Output behavior of auxiliary function group 9</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_TYPE[0]=&quot;M&quot;</td>
<td>; Description of auxiliary function 1: M40</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_EXTENSION[0]=0</td>
<td>; ... (and analogously for aux. functions 2 to 5)</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_VALUE[0]=40</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_GROUP[0]=9</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_TYPE[5]=&quot;M&quot;</td>
<td>; Description of auxiliary function 6: M45</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_EXTENSION[5]=0</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_VALUE[5]=45</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_GROUP[5]=9</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_TYPE[6]=&quot;M&quot;</td>
<td>; Description of auxiliary function 7: M1=40</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_EXTENSION[6]=1</td>
<td>; ... (and analogously for aux. functions 8 to 11)</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_VALUE[6]=40</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_GROUP[6]=9</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_TYPE[11]=&quot;M&quot;</td>
<td>; Description of auxiliary function 12: M1=45</td>
</tr>
<tr>
<td>Program code</td>
<td>Comment</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>$MN_AUXFU_GROUP_SPEC[9] = 'H22'</td>
<td>; Output behavior of auxiliary function group 10</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_TYPE[12] = &quot;M&quot;</td>
<td>; Description of auxiliary function 13: M2=3</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_EXTENSION[12] = 2</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_VALUE[12] = 3</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_GROUP[12] = 10</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_TYPE[13] = &quot;M&quot;</td>
<td>; Description of auxiliary function 14: M2=4</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_EXTENSION[13] = 2</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_VALUE[13] = 4</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_GROUP[13] = 10</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_TYPE[14] = &quot;M&quot;</td>
<td>; Description of auxiliary function 15: M2=5</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_EXTENSION[14] = 2</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_VALUE[14] = 5</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_GROUP[14] = 10</td>
<td></td>
</tr>
<tr>
<td>$MN_AUXFU_GROUP_SPEC[10] = 'H22'</td>
<td>; Specification of auxiliary function group 11</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_TYPE[16] = &quot;S&quot;</td>
<td>; Description of auxiliary function 17: S2=&lt;all values&gt;</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_EXTENSION[16] = 2</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_VALUE[16] = -1</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_GROUP[16] = 11</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_TYPE[17] = &quot;M&quot;</td>
<td>; Description of auxiliary function 18: M50</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_EXTENSION[17] = 0</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_VALUE[17] = 50</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_GROUP[17] = 12</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_TYPE[18] = &quot;M&quot;</td>
<td>; Description of auxiliary function 19: M51</td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_EXTENSION[18] = 0</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_VALUE[18] = 51</td>
<td></td>
</tr>
<tr>
<td>$MC_AUXFU_ASSIGN_GROUP[18] = 12</td>
<td></td>
</tr>
<tr>
<td>$MN_AUXFU_GROUP_SPEC[12] = 'H21'</td>
<td>; Specification of auxiliary function group 13</td>
</tr>
</tbody>
</table>
### 8.18 Data lists

#### 8.18.1 Machine data

##### 8.18.1.1 NC-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10713</td>
<td>M_NO_FCT_STOPRE</td>
<td>M function with preprocessing stop</td>
</tr>
<tr>
<td>10714</td>
<td>M_NO_FCT_EOP</td>
<td>M function for spindle active after NC RESET</td>
</tr>
<tr>
<td>10715</td>
<td>M_NO_FCT_CYCLE</td>
<td>M function to be replaced by subroutine</td>
</tr>
<tr>
<td>11100</td>
<td>AUXFU_MAXNUM_GROUP_ASSIGN</td>
<td>Maximum number of user-defined auxiliary functions per channel</td>
</tr>
<tr>
<td>11110</td>
<td>AUXFU_GROUP_SPEC</td>
<td>Group-specific output behavior</td>
</tr>
<tr>
<td>11450</td>
<td>SEARCH_RUN_MODE</td>
<td>Behavior after a block search</td>
</tr>
</tbody>
</table>

##### 8.18.1.2 Channelspecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20110</td>
<td>RESET_MODE_MASK</td>
<td>Definition of control initial setting after part program start.</td>
</tr>
<tr>
<td>20112</td>
<td>START_MODE_MASK</td>
<td>Definition of control initial setting after powerup and on RESET or at end of part program</td>
</tr>
<tr>
<td>20270</td>
<td>CUTTING_EDGE_DEFAULT</td>
<td>Basic setting of tool cutting edge without programming</td>
</tr>
<tr>
<td>20800</td>
<td>SPF_END_TO_VDI</td>
<td>Subroutine end / Stop at PLC</td>
</tr>
<tr>
<td>22000</td>
<td>AUXFU_ASSIGN_GROUP</td>
<td>Group assignment of user-defined auxiliary functions</td>
</tr>
<tr>
<td>22010</td>
<td>AUXFU_ASSIGN_TYPE</td>
<td>Type of user-defined auxiliary functions</td>
</tr>
<tr>
<td>22020</td>
<td>AUXFU_ASSIGN_EXTENSION</td>
<td>Address extension for user-defined auxiliary functions</td>
</tr>
</tbody>
</table>

---

```
Program code Comment
$MC_AUXFU_ASSIGN_TYPE[19]="M" ; Description of auxiliary function 20: M52
$MC_AUXFU_ASSIGN_EXTENSION[19]=0
$MC_AUXFU_ASSIGN_VALUE[19]=52
$MC_AUXFU_ASSIGN_GROUP[19]=13

$MC_AUXFU_ASSIGN_TYPE[20]="M" ; Description of auxiliary function 21: M53
$MC_AUXFU_ASSIGN_EXTENSION[20]=0
$MC_AUXFU_ASSIGN_VALUE[20]=53
$MC_AUXFU_ASSIGN_GROUP[20]=13
```
### 8.18.2 Signals

#### 8.18.2.1 Signals to channel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate associated M01</td>
<td>DB21, ... .DBX30.5</td>
<td>DB320x.DBX14.5</td>
</tr>
</tbody>
</table>

#### 8.18.2.2 Signals from channel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>M function 1 - 5 change</td>
<td>DB21, ... .DBX58.0-4</td>
<td>DB250x.DBX4.0-4</td>
</tr>
<tr>
<td>M function 1 - 5 not decoded</td>
<td>DB21, ... .DBX59.0-4</td>
<td>-</td>
</tr>
<tr>
<td>S function 1 - 3 change</td>
<td>DB21, ... .DBX60.0-2</td>
<td>DB250x.DBX6.0</td>
</tr>
<tr>
<td>S function 1 - 3 quick</td>
<td>DB21, ... .DBX60.4-6</td>
<td>-</td>
</tr>
<tr>
<td>T function 1 - 3 change</td>
<td>DB21, ... .DBX61.0-2</td>
<td>-</td>
</tr>
</tbody>
</table>
### H2: Auxiliary function outputs to PLC

#### 8.18 Data lists

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>T function 1 - 3 quick</td>
<td>DB21, ... DBX61.4-6</td>
<td>-</td>
</tr>
<tr>
<td>D function 1 - 3 change</td>
<td>DB21, ... DBX62.0-2</td>
<td>DB250x.DBX10.0</td>
</tr>
<tr>
<td>D function 1 - 3 quick</td>
<td>DB21, ... DBX62.4-6</td>
<td>-</td>
</tr>
<tr>
<td>DL function change</td>
<td>DB21, ... DBX63.0</td>
<td>-</td>
</tr>
<tr>
<td>DL function quick</td>
<td>DB21, ... DBX63.4</td>
<td>-</td>
</tr>
<tr>
<td>H function 1 - 3 change</td>
<td>DB21, ... DBX64.0-2</td>
<td>DB250x.DBX12.0-2</td>
</tr>
<tr>
<td>H function 1 - 3 quick</td>
<td>DB21, ... DBX64.4-6</td>
<td>-</td>
</tr>
<tr>
<td>F function 1 - 6 change</td>
<td>DB21, ... DBX65.0-5</td>
<td>-</td>
</tr>
<tr>
<td>M function 1 - 5 quick</td>
<td>DB21, ... DBX66.0-4</td>
<td>-</td>
</tr>
<tr>
<td>F function 1 - 6 quick</td>
<td>DB21, ... DBX67.0-5</td>
<td>-</td>
</tr>
<tr>
<td>Extended address M function 1 (16 bit int)</td>
<td>DB21, ... DBB68-69</td>
<td>DB250x.DBB3004</td>
</tr>
<tr>
<td>M function 1 (DInt)</td>
<td>DB21, ... DBB70-73</td>
<td>DB250x.DBD3000</td>
</tr>
<tr>
<td>Extended address M function 2 (16 bit int)</td>
<td>DB21, ... DBB74-75</td>
<td>DB250x.DBB3012</td>
</tr>
<tr>
<td>M function 2 (DInt)</td>
<td>DB21, ... DBB76-79</td>
<td>DB250x.DBD3008</td>
</tr>
<tr>
<td>Extended address M function 3 (16 bit int)</td>
<td>DB21, ... DBB80-81</td>
<td>DB250x.DBB3020</td>
</tr>
<tr>
<td>M function 3 (DInt)</td>
<td>DB21, ... DBB82-85</td>
<td>DB250x.DBD3016</td>
</tr>
<tr>
<td>Extended address M function 4 (16 bit int)</td>
<td>DB21, ... DBB86-87</td>
<td>DB250x.DBB3028</td>
</tr>
<tr>
<td>M function 4 (DInt)</td>
<td>DB21, ... DBB88-91</td>
<td>DB250x.DBD3024</td>
</tr>
<tr>
<td>Extended address M function 5 (16 bit int)</td>
<td>DB21, ... DBB92-93</td>
<td>DB250x.DBB3036</td>
</tr>
<tr>
<td>M function 5 (DInt)</td>
<td>DB21, ... DBB94-97</td>
<td>DB250x.DBD3032</td>
</tr>
<tr>
<td>Extended address S function 1 (16 bit int)</td>
<td>DB21, ... DBB98-99</td>
<td>DB250x.DBB4004</td>
</tr>
<tr>
<td>S function 1 (real)</td>
<td>DB21, ... DBB100-103</td>
<td>DB250x.DBD4000</td>
</tr>
<tr>
<td>Extended address S function 2 (16 bit int)</td>
<td>DB21, ... DBB104-105</td>
<td>DB250x.DBB4012</td>
</tr>
<tr>
<td>S function 2 (real)</td>
<td>DB21, ... DBB106-109</td>
<td>DB250x.DBD4008</td>
</tr>
<tr>
<td>Extended address S function 3 (16 bit int)</td>
<td>DB21, ... DBB110-111</td>
<td>DB250x.DBB4020</td>
</tr>
<tr>
<td>S function 3 (real)</td>
<td>DB21, ... DBB112-115</td>
<td>DB250x.DBD4016</td>
</tr>
<tr>
<td>Extended address T function 1 (16 bit int)</td>
<td>DB21, ... DBB116-117</td>
<td>-</td>
</tr>
<tr>
<td>T function 1 (integer)</td>
<td>DB21, ... DBB118-119</td>
<td>DB250x.DBD2000</td>
</tr>
<tr>
<td>Extended address T function 2 (16 bit int)</td>
<td>DB21, ... DBB120-121</td>
<td>-</td>
</tr>
<tr>
<td>T function 2 (integer)</td>
<td>DB21, ... DBB122-123</td>
<td>-</td>
</tr>
<tr>
<td>Extended address T function 3 (16 bit int)</td>
<td>DB21, ... DBB124-125</td>
<td>-</td>
</tr>
<tr>
<td>T function 3 (integer)</td>
<td>DB21, ... DBB126-127</td>
<td>-</td>
</tr>
<tr>
<td>Extended address D function 1 (8 bit int)</td>
<td>DB21, ... DBB128</td>
<td>-</td>
</tr>
<tr>
<td>D function 1 (8 bit int)</td>
<td>DB21, ... DBB129</td>
<td>DB250x.DBD5000</td>
</tr>
<tr>
<td>Extended address D function 2 (8 bit int)</td>
<td>DB21, ... DBB130</td>
<td>-</td>
</tr>
<tr>
<td>D function 2 (8 bit int)</td>
<td>DB21, ... DBB131</td>
<td>-</td>
</tr>
<tr>
<td>Extended address D function 3 (8 bit int)</td>
<td>DB21, ... DBB132</td>
<td>-</td>
</tr>
<tr>
<td>D function 3 (8 bit int)</td>
<td>DB21, ... DBB133</td>
<td>-</td>
</tr>
<tr>
<td>Extended address DL function (8 bit int)</td>
<td>DB21, ... DBB134</td>
<td>-</td>
</tr>
<tr>
<td>DL function (real)</td>
<td>DB21, ... DBB136</td>
<td>-</td>
</tr>
<tr>
<td>Extended address H function 1 (16 bit int)</td>
<td>DB21, ... DBB140-141</td>
<td>DB250x.DBB6004</td>
</tr>
<tr>
<td>Signal name</td>
<td>SINUMERIK 840D sl</td>
<td>SINUMERIK 828D</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>H function 1 (real or DInt)</td>
<td>DB21, ... DBB142-145</td>
<td>DB250x.DBD6000</td>
</tr>
<tr>
<td>Extended address H function 2 (16 bit int)</td>
<td>DB21, ... DBB146-147</td>
<td>DB250x.DBB6012</td>
</tr>
<tr>
<td>H function 2 (REAL or DInt)</td>
<td>DB21, ... DBB148-151</td>
<td>DB250x.DBD6008</td>
</tr>
<tr>
<td>Extended address H function 3 (16 bit int)</td>
<td>DB21, ... DBB152-153</td>
<td>DB250x.DBB6020</td>
</tr>
<tr>
<td>H function 3 (real or DInt)</td>
<td>DB21, ... DBB154-157</td>
<td>DB250x.DBD6016</td>
</tr>
<tr>
<td>Extended address F function 1 (16 bit int)</td>
<td>DB21, ... DBB158-159</td>
<td>-</td>
</tr>
<tr>
<td>F function 1 (real)</td>
<td>DB21, ... DBB160-163</td>
<td>-</td>
</tr>
<tr>
<td>Extended address F function 2 (16 bit int)</td>
<td>DB21, ... DBB164-165</td>
<td>-</td>
</tr>
<tr>
<td>F function 2 (real)</td>
<td>DB21, ... DBB166-169</td>
<td>-</td>
</tr>
<tr>
<td>Extended address F function 3 (16 bit int)</td>
<td>DB21, ... DBB170-171</td>
<td>-</td>
</tr>
<tr>
<td>F function 3 (real)</td>
<td>DB21, ... DBB172-175</td>
<td>-</td>
</tr>
<tr>
<td>Extended address F function 4 (16 bit int)</td>
<td>DB21, ... DBB176-177</td>
<td>-</td>
</tr>
<tr>
<td>F function 4 (real)</td>
<td>DB21, ... DBB178-181</td>
<td>-</td>
</tr>
<tr>
<td>Extended address F function 5 (16 bit int)</td>
<td>DB21, ... DBB182-183</td>
<td>-</td>
</tr>
<tr>
<td>F function 5 (real)</td>
<td>DB21, ... DBB184-187</td>
<td>-</td>
</tr>
<tr>
<td>Extended address F function 6 (16 bit int)</td>
<td>DB21, ... DBB188-189</td>
<td>-</td>
</tr>
<tr>
<td>F function 6 (real)</td>
<td>DB21, ... DBB190-193</td>
<td>-</td>
</tr>
<tr>
<td>Dynamic M function: M00 - M07</td>
<td>DB21, ... DBB194</td>
<td>DB250x.DBB1000</td>
</tr>
<tr>
<td>Dynamic M function: M08 - M15</td>
<td>DB21, ... DBB195</td>
<td>DB250x.DBB1001</td>
</tr>
<tr>
<td>Dynamic M function: M16 - M23</td>
<td>DB21, ... DBB196</td>
<td>DB250x.DBB1002</td>
</tr>
<tr>
<td>Dynamic M function: M24 - M31</td>
<td>DB21, ... DBB197</td>
<td>DB250x.DBB1003</td>
</tr>
<tr>
<td>Dynamic M function: M32 - M39</td>
<td>DB21, ... DBB198</td>
<td>DB250x.DBB1004</td>
</tr>
<tr>
<td>Dynamic M function: M40 - M47</td>
<td>DB21, ... DBB199</td>
<td>DB250x.DBB1005</td>
</tr>
<tr>
<td>Dynamic M function: M48 - M55</td>
<td>DB21, ... DBB200</td>
<td>DB250x.DBB1006</td>
</tr>
<tr>
<td>Dynamic M function: M56 - M63</td>
<td>DB21, ... DBB201</td>
<td>DB250x.DBB1007</td>
</tr>
<tr>
<td>Dynamic M function: M64 - M71</td>
<td>DB21, ... DBB202</td>
<td>DB250x.DBB1008</td>
</tr>
<tr>
<td>Dynamic M function: M72 - M79</td>
<td>DB21, ... DBB203</td>
<td>DB250x.DBB1009</td>
</tr>
<tr>
<td>Dynamic M function: M80 - M87</td>
<td>DB21, ... DBB204</td>
<td>DB250x.DBB1010</td>
</tr>
<tr>
<td>Dynamic M function: M88 - M95</td>
<td>DB21, ... DBB205</td>
<td>DB250x.DBB1011</td>
</tr>
<tr>
<td>Dynamic M function: M96 - M99</td>
<td>DB21, ... DBX206.0-3</td>
<td>DB250x.DBB1012.0-3</td>
</tr>
<tr>
<td>Associated M00/M01 active</td>
<td>DB21, ... DBX318.5</td>
<td>DB330x.DBX402.5</td>
</tr>
</tbody>
</table>

### 8.18.2.3 Signals to axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>F function for positioning axis (real)</td>
<td>DB31, ... DBB78-81</td>
<td>-</td>
</tr>
</tbody>
</table>
### 8.18.2.4 Signals from axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>M function for spindle (Int)</td>
<td>DB21, … .DBB86-87</td>
<td>DB370x.DB0000</td>
</tr>
<tr>
<td>S function for spindle (real)</td>
<td>DB21, … .DBB88-91</td>
<td>DB370x.DBD0004</td>
</tr>
</tbody>
</table>
9.1 Product brief

Channel
An NC channel represents the smallest unit for manual traversing of axes and automatic processing of part programs. At any one time, a channel will always be in a particular mode, e.g. AUTOMATIC, MDI, or JOG. A channel can be regarded as an independent NC.

Mode group
A channel always belongs to a mode group. A mode group can also consist of several channels. A mode group can be identified by the fact that all channels of the mode group are always in the same mode at a particular time, e.g. AUTOMATIC, MDI, or JOG. This is ensured through the NC internal mode logic.

A mode group can be regarded as an independent multi-channel NC.

Channel gaps
When channels are configured, placeholder channels can be provided in order to create as uniform a configuration as possible over machines in a series. Only the channels that are actually used are then activated.

Program test
The following options are available for testing or moving in position a new part program.
- Program execution without setpoint outputs
- Program execution in singleblock mode
- Program execution with dry run feedrate
- Skip part program blocks
- Block search with or without calculation.

Block search
The block search function enables the following program simulations for locating specific program points:
- Type 1 without calculation at contour
- Type 2 with calculation at contour
Program operation

The execution of part programs or part program blocks in AUTOMATIC or MDI modes is referred to as program operation. During execution, the program sequence can be controlled by PLC interface signals and commands.

For each channel, basic settings or channel-specific machine data can be specified. These initial settings affect, for example, G groups and auxiliary function output.

A part program can be selected only if the relevant channel is in the Reset state.

Furthermore, all further program runs are handled by PLC interface signals and the corresponding commands.

- Start of part program or part program block
- Part program calculation and program control
- RESET command, program status, and channel status
- Responses to operator and program actions
- Event-driven program calls

Asynchronous subroutines (ASUBs), interrupt routines

Interrupt inputs allow the NC to interrupt the current part program execution so that it can react to more urgent events in interrupt routines or ASUBs.

Single block

With the single-block function, the user can execute a part program block-by-block.

There are 3 types of setting for the single-block function:

- SLB1: = IPO single block
- SLB2: = Decode single block
- SLB3: = Stop in cycle

Basic block display

A second basic block display can be used with the existing block display to display all blocks that produce an action on the machine.

The actually approached end positions are shown as an absolute position. The position values refer either to the workpiece coordinate system (WCS) or the settable zero system (SZS).
Program execution from external source

When complex workpieces are machined, the NC may not have enough memory for the programs. Using the function "Execution from external source" or "EES (execution from external storage) enables part programs to be called and executed from an external memory (e.g. from the hard disk).

Behavior after POWER ON, Reset, ...

The control-system response after:
- Power up (POWER ON)
- Reset/part program end
- Part program start

can be modified for functions, such as G codes, tool length compensation, transformation, coupled axis groupings, tangential follow-up, programmable synchronous spindle for certain system settings through machine data.

Subroutine call with M, T and D functions

For certain applications, it may be advantageous to replace M, T or D functions as well as a few NC language commands SPOS, SPOSA, by a subroutine call. This can be used, for example, to call the tool change routine.

Relevant machine data can be used to define and control subroutines having M, T or D functions. For example, for a gear stage change.

Program runtime/part counter

Information on the program runtime and the part count is provided to assist the machine tool operator.

The functions defined for this purpose are not identical to the functions of tool management and are intended primarily for systems without tool management.

9.2 Mode group

Mode group

In a mode group, multiple channels if an NC are grouped together to form one processing unit. In principle, it is an independent "NC" within an NC.
A mode group is essentially characterized by the fact that all channels assigned to it are any instant are always in the same mode (AUTOMATIC, JOG, MDI).

**Note**
This description continues on the assumption that there is one mode group and one channel. Functions that need several channels, e.g. "Axis replacement" function, are described in: **References:**
Function Manual Extension functions; Mode Groups, Channels, Axis replacement (K5)

---

### Parameter assignment

A channel is assigned to a mode group in the machine data:

MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP[<channel index>] = mode group number

All channels with the same mode group number form one mode group.

**Note**
A NC always comprises one mode group with one channel by default. It is not possible to parameterize an NC without a channel.

---

### Mode-group-specific NC/PLC interface signals

The mode-group-specific NC/PLC interface signals are in data block DB11.
The mode-group-specific NC/PLC interface essentially comprises the following interface signals:

- **Request signals PLC → NC**
  - Mode group reset
  - Mode group stop axes plus spindles
  - Mode group stop
  - Mode changeover inhibit
  - Mode: JOG, MDA, AUTOM.
  - Single block: Type A, Type B
  - Machine function REF, REPOS, TEACH IN, INC x

- **Status signals NC → PLC**
  - Mode strobe: JOG, MDA, AUTOMATIC
  - Machine function strobe: REF, REPOS, TEACH IN
  - All channels in the reset state
  - Mode group has been reset
  - Mode group Ready
  - Active mode: JOG, MDA, AUTOMATIC
  - Active machine function: REF, REPOS, TEACH, INC x

### Activated and non-activated channels, channel gaps

**Activated channel**

A channel with a mode group number ≠ 0 is an active channel.

**Non-activated channel**

A channel with a mode group number 0 is a non-activated channel. It does not take up any memory internally in the control.

**Channel gaps**

A channel with a mode group number 0 is not only a non-activated channel. It is also a so-called channel gap in the sequence of the channels.

The advantage of channel gaps is that the configuration data can be kept largely identical in a series of similar machines. For the specific commissioning, only those channels are activated
that are required for the machine in question. The unoccupied memory can be freely used as additional user memory.

Table 9-1 Example

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD10010 $MN_AS-SIGN_CHAN_TO_MODE_GROUP[0] = 1</td>
<td>Channel 1, mode group 1</td>
</tr>
<tr>
<td>MD10010 $MN_AS-SIGN_CHAN_TO_MODE_GROUP[1] = 2</td>
<td>Channel 2, mode group 2</td>
</tr>
<tr>
<td>MD10010 $MN_AS-SIGN_CHAN_TO_MODE_GROUP[2] = 0</td>
<td>Channel 3, not used</td>
</tr>
<tr>
<td>MD10010 $MN_AS-SIGN_CHAN_TO_MODE_GROUP[3] = 1</td>
<td>Channel 4, mode group 1</td>
</tr>
<tr>
<td>MD10010 $MN_AS-SIGN_CHAN_TO_MODE_GROUP[4] = 2</td>
<td>Channel 5, mode group 2</td>
</tr>
</tbody>
</table>

See also

K2: Axis Types, Coordinate Systems, Frames (Page 661)
S1: Spindles (Page 1199)

9.2.1 Mode group stop

Function

The following NC/PLC interface signals are used to stop the traversing motions of the axes or of the axes and spindles in all mode group channels and to interrupt part program execution:

DB11 DBX0.5 (mode group stop)
DB11 DBX0.6 (mode group stop, axes plus spindles)

9.2.2 Mode group reset

Function

A mode group Reset is requested via a mode group-specific NC/PLC interface signal:

DB11 DBX0.7 = 1 (mode group reset)
Effect on the channels of mode group:

- Part program preparation (preprocessing) is stopped.
- All axes and spindles are decelerated to zero speed according to their acceleration curves without contour violation.
- Any auxiliary functions not yet output to the PLC are no longer output.
- The preprocessing pointers are set to the interruption point, and the block pointers are set to the beginning of the appropriate part programs.
- All initial settings (e.g. the G functions) are set to the parameterized values.
- All alarms with "Channel reset" criterion are canceled.

If all the channels of the mode group are in reset state, then:

- All alarms with "Mode group reset" cancel criterion are canceled.
- The NC/PLC interface indicates completion of the mode group Reset and the mode group's readiness to operate:
  - DB11 DBX6.7 (all channels in the reset state)
  - DB11 DBX6.3 = 1 (mode group ready)

9.3 Mode types and mode type change

Unique mode

All channels of a mode group are always in the same mode:

- AUTOMATIC
- JOG
- MDI

If individual channels are assigned to different mode groups, a channel switchover activates the corresponding mode group. This allows mode changes to be initiated via a channel switchover.

Modes

The following modes are available:

- AUTOMATIC
  Automatic execution of part programs:
  - Part program test.
  - All channels of the mode group can be active at the same time.

- JOG in Automatic
  JOG in AUTOMATIC is an extension of AUTOMATIC mode intended to simplify use. JOG can be executed without leaving AUTOMATIC mode if boundary conditions so permit.
• **JOG**
  Manual traversing of axes via traversing keys of the machine control panel or via a handwheel connected to the machine control panel:
  – Channel-specific signals and interlocks are taken into account for motions executed by means of an ASUB or via static synchronized actions.
  – Couplings are taken into account.
  – Every channel in the mode group can be active.

• **MDI**
  Manual Data Automatic (the blocks are entered via the user interface)
  – Restricted execution of part programs and part program sections.
  – Part program test.
  – A maximum of one channel per mode group can be active (applies only to TEACH IN).
  – Axes can only be traversed manually in subordinate machine functions such as JOG, REPOS or TEACH IN.

**Applies to all modes**

**Cross-mode synchronized actions**

Modal synchronized actions can be executed by means of IDS in all modes for the following functions parallel to the channel:

- Command axis functions
- Spindle functions
- Technology cycles

**Selection**

The user can select the desired mode by means of softkeys on the user interface.

This selection (AUTOMATIC, MDI, or JOG) is forwarded to the PLC on the NC/PLC interface, but is not activated:

DB11 DBX4.0, 0.1, 0.2 (strobe mode)

**Activation and priorities**

The mode of the mode group is activated via the NC/PLC interface:

DB11 DBX0.0, 0.1, 0.2 (mode)

If several modes are selected at the same time, the following priority applies:

<table>
<thead>
<tr>
<th>Priority</th>
<th>Mode</th>
<th>Mode-group signal (NCK → PLC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st priority, high</td>
<td>JOG</td>
<td>DB11 DBX0.2</td>
</tr>
<tr>
<td>2nd priority, medium</td>
<td>MDI</td>
<td>DB11 DBX0.1</td>
</tr>
<tr>
<td>3rd priority, low</td>
<td>AUTOMATIC</td>
<td>DB11 DBX0.0</td>
</tr>
</tbody>
</table>
Display

The current mode of the mode group is displayed via the NC/PLC interface:

DB11 DBX6.0, 0.1, 0.2 (active mode)

<table>
<thead>
<tr>
<th>Mode-group signal (NCK → PLC)</th>
<th>Active operating mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB11 DBX6.2</td>
<td>JOG</td>
</tr>
<tr>
<td>DB11 DBX6.1</td>
<td>MDI</td>
</tr>
<tr>
<td>DB11 DBX6.0</td>
<td>AUTOMATIC</td>
</tr>
</tbody>
</table>

Machine functions

Machine functions can be selected within a mode that also apply within the mode group:

- Machine functions within the JOG mode
  - REF (reference point approach)
  - REPOS (repositioning)
  - JOG retract (retraction motion in the tool direction)

- Machine functions within the MDI mode
  - REF (reference point approach)
  - REPOS (repositioning)
  - TEACHIN (teach-in of axis positions)

NC/PLC interface signals

- DB11 DBX5.0, 0.1, 0.2 (machine function strobe): Requirement
- DB11 DBX1.0, 0.1, 0.2 (machine function): Activation
- DB11 DBX7.0-2 (active machine function): Feedback signal

Channel states

- **Channel reset**
  The machine is in its initial state. This is defined by the machine manufacturer's PLC program, e.g. after POWER ON or at the end of the program.

- **Channel active**
  A program has been started, and the program is being executed or a reference point approach is in progress.

- **Channel interrupted**
  The running program or reference point approach has been interrupted.

Functions within modes

Modes are supplemented through user-specific functions. The individual functions are technology and machine-independent and can be started and/or executed from the three channel states "Channel reset", "Channel active" or "Channel interrupted".
Supplementary condition for submode TEACH IN

TEACH IN is not permissible for leading or following axes of an active axis grouping, e.g. for:

- Gantry-axis grouping or a gantry axis pair
- Coupled-axis grouping of master and slave axis

JOG in Automatic

JOG in AUTOMATIC mode is permitted if the mode group is in "RESET" state and the axis is jog-capable. "RESET" state for the mode group means:

- All channels are in the "RESET" state
- All programs are canceled
- DRF is not active in any channel

An axis is **JOG-capable** if it is not in any of the following states:

- PLC axis as concurrent positioning axis (request of the axis from the PLC)
- Command axis (the axis has been programmed by a synchronized action and the motion has not been completed yet)
- Rotating spindle (spindle rotating despite RESET)
- An asynchronous reciprocating axis

Note: The “jog-capable” property is independent of the “JOG in AUTOMATIC” function.

Activation

The function "JOG in AUTOMATIC" can be activated with the machine data:

MD10735 $MN_JOG_MODE_MASK

- Before POWER ON, the following machine data must be set:
  MD10735 $MN_JOG_MODE_MASK, bit 0 = 1
- The user switches to AUTO (PLC user interface DB11 DBX0.0 = 0→1 edge). “JOG in AUTOMATIC” is then active if the NCK previously had channel state “RESET” and program state “Aborted” in all mode group channels. The axis in question must also be “jog-capable”. DRF must be deactivated (if not already deactivated).
- RESET is initiated or the running program is finished with M30/M2 in all mode group channels that do not have channel state “Reset” and program state “Aborted”.
- The relevant axis is automatically made “JOG-capable” (e.g. axis interchange: PLC → NC).

Note: In most applications, the axes to be traversed are “JOG-capable” and with the switchover to AUTOMATIC, "JOG in AUTOMATIC" is also active.

Characteristics

- The +/- keys cause a JOG movement, and the mode group is switched internally to JOG. (i.e. “Internal JOG”).
- Moving the handwheels causes a JOG movement, and the mode group is switched internally to JOG, unless DRF is active.
- An ongoing JOG movement is not complete until the end position of the increment has been reached (if this has been set) or the movement has been aborted with “Delete distance-to-go”.
  In this way an increment can be stopped using Stop and then moved to the end using Start.
  The NCK remains in “Internal JOG” during this time. A partial increment is possible, but it must not be interrupted using Stop. There is a mode in which releasing the travel key causes interruption within an increment.

- Without any JOG movement, “JOG in AUTOMATIC” responds in the same way as “Automatic”. In particular, the Start key starts the selected part program and the appropriate HMI softkey initiates a block search.

- If JOG movement is active, the NCK is internally in JOG mode, and, thus, a block search request is refused and a Start cannot start the part program. Start starts any remaining increment or has no effect.

- While a mode group axis is being traversed in JOG mode, the mode group remains internally in JOG mode.
  Comment: This phase can begin with the JOG movement of an axis and end with the end of the JOG movement of another axis.

- Axis interchange is not possible for an axis with active JOG movement (the axis might change mode group). The NCK blocks any axis interchange attempt.

- The PLC user interface indicates “Automatic” mode:
  - DB11 DBX6.0, 6.1, 6.2 = 1
  - DB11 DBX7.0, 7.1, 7.2 = 0

- In “JOG in AUTOMATIC”, the NC/PLC interface displays whether the mode group is in “Mode group RESET”.
  - DB11 DBX6.4 (mode group has been reset, mode group 1)
  - DB11 DBX26.4 (mode group has been reset, mode group 2)
  - DB11 DBX46.4 (mode group has been reset, mode group 3)

- In “JOG in AUTOMATIC”, the NC/PLC interface displays whether the NC has automatically switched to “Internal JOG”.
  - DB11 DBX6.5 (NCK internal JOG active, mode group 1)
  - DB11 DBX26.5 (NCK internal JOG active, mode group 2)
  - DB11 DBX46.5 (NCK internal JOG active, mode group 3)

**Supplementary conditions**

“JOG in AUTOMATIC” can only switch internally to JOG if the mode group is in “Mode group RESET” state, i.e. it is not possible to jog immediately in the middle of a stopped program. The user can jog in this situation by pressing the JOG key or the Reset key *in all channels* of the mode group.

Selecting AUTOMATIC disables the INC keys and the user can/must press the INC keys again to select the desired increment. If the NCK switches to “Internal JOG”, the selected increment is retained.

If the user attempts to jog the geometry or orientation axes, the NCK switches to “Internal JOG” and the movement executed. Several axes can be physically moved in this way; they must all be “JOG-capable”.

Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2

465
Following the JOG movement, the NCK deactivates “Internal JOG” again and selects AUTO mode again. The internal mode change is delayed until the movement is complete. This avoids unnecessary multiple switching operations, e.g. when using the handwheel. The PLC can only rely on the “Internal JOG active” PLC signal.

The NCK will then switch to “Internal JOG” if the axis is not enabled.

See also

R1: Referencing (Page 1149)

9.3.1 Monitoring functions and interlocks of the individual modes

Channel status determines monitoring functions

Monitoring functions in operating modes
Different monitoring functions are active in individual operating modes. These monitoring functions are not related to any particular technology or machine.

In a particular mode only some of the monitoring functions are active depending on the operating status. The channel status determines which monitoring functions are active in which mode and in which operating state.

Interlocking functions in operating modes
Different interlocks can be active in the different operating modes. These interlocking functions are not related to any particular technology or machine.

Almost all the interlocks can be activated in every mode, depending on the operating status.

9.3.2 Mode change

Introduction
A mode change is requested and activated via the mode group interface (DB11, ...). A mode group will either be in AUTOMATIC, JOG, or MDA mode, i.e. it is not possible for several channels of a mode group to take on different modes at the same time.

What mode transitions are possible and how these are executed can be configured in the PLC program on a machine-specific basis.

Note
The mode is not changed internally until the signal “Channel status active” is no longer pending. For error-free mode change however, all channels must assume a permissible operating mode.
Possible mode changes

The following table shows possible mode changes for one channel.

<table>
<thead>
<tr>
<th>AUTOMATIC</th>
<th>JOG</th>
<th>MDA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AUTO</td>
<td>MDA</td>
</tr>
<tr>
<td>Reset</td>
<td>Interrupt</td>
<td>Reset</td>
</tr>
<tr>
<td>AUTOMATIC</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>JOG</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MDA</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Possible mode changes are shown by an "X".

Special cases

- **Errors during mode change**
  If a mode change request is rejected by the system, the error message "Operating mode cannot be changed until after NC Stop" is output. This error message can be cleared without changing the channel status.

- **Mode change disable**
  A mode change can be prevented by means of interface signal: DB11, DBX0.4 (Mode change disable).
  This suppresses the mode change request. The user must configure a message to the operator indicating that mode change is disabled. No signal is set by the system.

- **Mode change from MDA to JOG**
  If all channels of the mode group are in Reset state after a mode change from MDA to JOG, the NC switches from JOG to AUTO. In this state, part program commands START or INIT can be executed.
  If a channel of the mode group is no longer in Reset state after a mode change, the part program command START is rejected and Alarm 16952 is issued.

9.4 Channel

In a channel of the NC, the part programs defined by the user are executed.

The following properties characterize a channel:

- A channel is always assigned to a mode group (Page 457) (BAG).
- A channel can only execute one part program at any point in time.
- Machine, geometry, and special axes and spindles are assigned to a channel. Only these can be traversed via the part program executed in the channel.
- A channel consists of the internal units
  - Preprocessing (program decoding and block preparation)
  - Main run (path and axis interpolation).
A channel has an interface with the PLC. Via this NC/PLC interface, the PLC user program can read various channel-specific status data and write requests to the channel.

- Channel-specific tool offsets (Page 1363) are active in a channel.
- Channel-specific coordinate systems (Page 681) are active in a channel.
- Each channel has a unique channel name by default. The defined channel name can be changed in machine data:
  \[ \text{MD20000 } \$\text{MC\_CHAN\_NAME} = "\text{<channel name>}" \]

Multiple channels can be combined to form a mode group. The channels of a mode group are always in the same mode (AUTOMATIC, JOG, MDI).

**Channel configuration**

Channels can be filled with their own channel name via the following machine data:

\[ \text{MD20000 } \$\text{MC\_CHAN\_NAME} \text{ (channel name)} \]

The various axes are then assigned to the available channels via machine data. There can be only one setpoint-issuing channel at a time for an axis/spindle. The axis/spindle actual value can be read by several channels at the same time. The axis/spindle must be registered with the relevant channel.

The following channel-specific settings can also be made using machine data:

- Position of deletions or the basic program settings of G groups via the machine data:
  \[ \text{MD20150 } \$\text{MC\_GCODE\_RESET\_VALUES} \text{ (initial setting of the G groups)} \]
- Auxiliary function groups regarding the combination and the output time.
- Transformation conditions between machine axes and geometry axes.
- Other settings for the execution of a part program.

**Change in the channel assignment**

An online change in the channel configuration cannot be programmed in a part program or PLC user program. Changes in the configuration must be made via the machine data. The changes become effective only after a new POWER ON.

**Container axes and link axes**

An axis container combines a group of axes in a container. These axes are referred to as container axes. This involves assigning a pointer to a container slot (ring buffer location within the relevant container) to a channel axis. One of the axes in the container is located temporarily in this slot.

Each machine axis in the axis container must be assigned at all times to exactly one channel axis.

Link axes can be assigned permanently to one channel or dynamically (by means of an axis container switch) to several channels of the local NCU or the other NCU. A link axis is a non-local axis from the perspective of one of the channels belonging to the NCU to which the axis is not physically connected.
The assignment between the link axes and a channel is implemented as follows:

- For permanent assignment using machine data:
  Allow the direct logic machine axis image to show link axes.
- For dynamic assignment:
  Allow the axis container slot machine data to show link axes.

Further information on link axes and container axes can be found in:

**References:**
Function Manual, Extended Functions; Several Operator Panel Fronts and Multiple NCUs, Distributed Systems (B3)

### Interface signals

The signals of the 1st channel are located in the NC/PLC interface in DB21, the signals from channel 2 are located in DB22. The channel or channels can be monitored and controlled from the PLC or NCK.

### Channel-specific technology specification

The technology used can be specified for each channel:

```
MD27800 $MC_TECHNOLOGY_MODE
```

In the delivery state, machine data is active for milling as standard.

### Spindle functions using a PLC

In addition to function block FC18, spindle functions can also be started and stopped via the axial NC/PLC interface signals in parallel to part programs that are running.

**Requirements:**

- Channel status: "Interrupted" or "RESET"
- Program status: "Interrupted" or "canceled"

The following functions can be controlled from the PLC via interface signals:

- Stop (corresponds to \texttt{M5})
- Start with clockwise direction of rotation (corresponds to \texttt{M3})
- Start with counter-clockwise direction of rotation (corresponds to \texttt{M4})
- Select gear stage
- Positioning (corresponds to \texttt{M19})

For several channels, the spindle started by the PLC is active in the channel to which it is assigned at the start.

For more information on the special spindle interface, see Section "S1: Spindles (Page 1199)."
PLC-controlled single-axis operations

An axis can also be controlled from the PLC instead of from a channel. For this purpose, the PLC requests the axis from the NC via the NC/PLC interface:

DB31, ... DBX28.7 = 1 (PLC controls axis)

The following functions can be controlled from the PLC:

- Cancel axis/spindle sequence (equivalent to delete distance-to-go)
- Stop or interrupt axis/spindle
- Resume axis/spindle operation (continue the motion sequence)
- Reset axis/spindle to the initial state

For more information on the channel-specific signal exchange (PLC → NCK), see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 821)."

The exact functionality of independent single-axis operations is described in:

References:
Function Manual, Extended Functions; Positioning Axes (P2)

9.4.1 Global start disable for channel

User/PLC

A global Start disable can be set for the selected channel via the HMI or from the PLC.

Function

When Start disable is set, no new program starts are accepted for the selected channel. Start attempts are counted internally.

If a start is executed by the PLC before a global block disable is sent from the HMI to the NCK, the program is not stopped by the Start disable and its status is transmitted to the HMI.

NC Start disable and global Start disable have the same effect on the internal counter for starts that have been sent but not executed. (OPI variable startRejectCount).

Bypassing global Start disable

The interface signal:
DB21, ... DBX7.5 (PLC → NCK)
allows the PLC to temporarily bypass a global Start disable.

0: Global Start disable is effective
1: Global Start disable is temporarily canceled.
Messages

If desired, a message can be issued when a Start attempt occurs while a global block disable is active.

The control is exercised using machine data:

MD11411 $MN_ENABLE_ALARM_MASK Bit 6

1: Alarm 16956 appears: Channel %1, Program %2 cannot be started because of "Global Start disable".
0: Start attempts when a global block disable is set are not signaled by an alarm.

9.5 Program test

Several control functions are available for testing a new part program. These functions are provided to reduce danger at the machine and time required for the test phase. Several program functions can be activated at the same time to achieve a better result.

Test options

The following test options are described below:

- Program execution without setpoint outputs
- Program execution in singleblock mode
- Program execution with dry run feedrate
- Skip part program blocks
- Block search with or without calculation.

9.5.1 Program execution without setpoint outputs

Function

In the "Program test" status, a part program is executed without the output of axis or spindle setpoints.

The user can use this to check the programmed axis positions and auxiliary function outputs of a part program. This program simulation can also be used as an extended syntax check.

Selection

This function is selected via the operator interface in the "Program control" menu.

The selection sets the following interface signal:

DB21, ... DBX25.7 (program test selected)

This does not activate the function.
Activation

The function is activated via interface signal:
DB21, ... DBX1.7 (activate program test)

Display

The corresponding field on the operator interface is reversed and the interface signal in the PLC as a checkback of the active program test:
DB21, ... DBX33.7 (program test active)

Program start and program run

When the program test function is active, the part program can be started and executed (incl. auxiliary function outputs, wait times, G function outputs etc.) via the interface signal:
DB21, ... DBX7.1 (NC-Start)

The safety functions such as software limit switch, working area limits continue to be valid. The only difference compared to normal program operation is that an internal axis disable is set for all axes (including spindles). The machine axes do not move, the actual values are generated internally from the setpoints that are not output. The programmed velocities remain unchanged. This means that the position and velocity information on the operator interface is exactly the same as that output during normal part program execution. The position control is not interrupted when this function is active, so the axes do not have to be referenced when the function is switched off.

⚠️ CAUTION

The signals for exact stop:
DB31, ... DBX60.6/60.7 (exact stop coarse/fine)
mirror the actual status on the machine.

They are only canceled during program testing if the axis is pushed out of its set position (the set position remains constant during program testing).

With signal:
DB21, ... DBX33.7 (program test active)
both the PLC program and the part program can use variable $P_ISTEST to decide how to react or branch in response to these signals during testing.

Note

Dry run feedrate

"Program execution without axis motion" can also be activated with the function "Dry run feedrate". With this function, part program sections with a small programmed feedrate can be processed in a shorter time.
Note

Tool management

Because of the axis disable, the assignment of a tool magazine is not changed during program testing. A PLC application must be used to ensure that the integrity of the data in the tool management system and the magazine is not corrupted. The toolbox diskettes contain an example of the basic PLC program.

9.5.2 Program execution in single-block mode

Function

In case of "Program execution in single-block mode" the part program execution stops after every program block. If tool cutter radius compensation or a tool nose radius correction is selected, processing stops after every intermediate block inserted by the controller.

The program status switches to "Program status stopped".

The channel status remains active.

The next part program block is processed on NC Start.

Application

The user can execute a part program block-by-block to check the individual machining steps. Once the user decides that an executed part program block is functioning correctly, the next block can be called.

Single-block types

The following single block types are differentiated:

- Decoding single block
  With this type of single block, all blocks of the part program (even the pure computation blocks without traversing motions) are processed sequentially by "NC Start".

- Action single block (initial setting)
  With this type of single block, the blocks that initiate actions (traversing motions, auxiliary function outputs, etc.) are processed individually.
  Blocks that were generated additionally during decoding (e.g. for cutter radius compensation at acute angles) are also processed individually in single-block mode.
  Processing is however not stopped at calculation blocks as these do not trigger actions.
The single-block types are determined via the user interface in the menu "Program controls".

<table>
<thead>
<tr>
<th>CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Function feature for single-block type series</strong></td>
</tr>
<tr>
<td>In a series of G33/G34/G35 blocks, a single block is only operative if &quot;dry run feed&quot; is selected. Calculation blocks are not processed in single-step mode (only if single decoding block is active). SBL2 is also ineffective with G33/G34/G35.</td>
</tr>
</tbody>
</table>

### Selection

It is possible to select the single-block mode:

- Via the machine control panel (key "Single Block")
- Via the user interface
  
  For an exact procedure, see:

  **References:**
  
  Operating Manual of the installed HMI application

### Activation

The function is activated through the PLC basic program via the interface signal:

DB21, ... DBX0.4 (activate single block)

### Display

Active single-block mode is indicated by a reversal in the relevant field in the status line on the user interface.

Because of the single-block mode, as soon as the part program processing has processed a part program block, the following interface signal is set:

DB21, ... DBX35.3 (program status interrupted)

### Processing without single-block stop

Despite the selected single-block mode, a processing without the single-block stop can be set for specific program runs, e.g. for:

- Internal ASUBs
- User ASUBs
- Intermediate blocks
- Block search group blocks (action blocks)
- Init blocks
- Subprograms with DISPLOF
9.5 Program test

9.5.3 Program execution with dry run feedrate

Function

During "Program execution with dry run feedrate" the traversing speeds which have been programmed together with G01, G02, G03, G33, G34 and G35, are replaced by a parameterized feedrate value:

SD42100 $SC_DRY_RUN_FEED (dry run feedrate)

The dry run feedrate also replaces the programmed revolutional feedrate in program blocks with G95.

The exact way of working of the parameterized dry run feedrate (SD42100) depends on the setting of another setting data:

SD42101 $SC_DRY_RUN_FEED_MODE (mode for test run speed)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0     | Dry run feedrate is the maximum of the programmed feedrate and setting data SD42100. (Default setting!)  
⇒ SD42100 becomes effective only when the stored value is greater than the programmed feedrate. |
| 1     | Dry run feedrate is the minimum of the programmed feedrate and SD42100.  
⇒ SD42100 becomes effective only when the stored value is less than the programmed feedrate. |
| 2     | The value in SD42100 acts as the dry run feedrate, regardless of the programmed feedrate. |
| 10    | As in the case of "0", except thread cutting (G33, G34, G35) and thread boring (G331, G332, G63). These functions are executed as programmed. |
| 11    | As in the case of "1", except thread cutting (G33, G34, G35) and thread boring (G331, G332, G63). These functions are executed as programmed. |
| 12    | As in the case of "2", except thread cutting (G33, G34, G35) and thread boring (G331, G332, G63). These functions are executed as programmed. |

A dry run feedrate can be selected in the automatic modes and activated on interruption of an automatic mode or end of a block.

For further information on influencing the feedrate, see Section "V1: Feedrates (Page 1307)".

References:
List Manual, Detailed Description of the Machine Data
Application

⚠️ DANGER

High cutting speeds

Workpieces may not be machined when "dry run feedrate" is active because the altered feedrates might cause the permissible tool cutting rates to be exceeded and the workpiece or machine tool could be damaged.

Selection

This function is selected via the user interface in the "Program control" menu.

The selection sets the following interface signal:

DB21, ... DBX24.6 (dry run feedrate selected)

This does not activate the function.

Activation

The function is activated via interface signal:

DB21, ... DBX0.6 (activate dry run feed)

Display

Active dry run feedrate mode is indicated by a reversal in the relevant field in the status line on the user interface.
9.5.4 Skip part-program blocks

Function

When testing or breaking in new programs, it is useful to be able to disable or skip certain part program blocks during program execution. For this, the respective records must be marked with a slash.

Main program/subroutine
/
N40 ...
N50 ...
N60 ...
N70 ...
N80 ...
N90 ...
N100 ...
N110 ...
N120 M30

Figure 9-1 Skipping part program blocks

Selection

This function is selected via the operator interface in the "Program control" menu.
The selection sets the following interface signal:
DB21, ... DBX26.0 (skip block selected)
This does not activate the function.

Activation

The function is activated via the interface signal:
DB21, ... DBX2.0 (activate skip block)

Note

The "Skip part programs" function remains active during block searches.
Display

Activated "Skip block" function is indicated by a reversal of the relevant field on the operator interface.

9.6 Workpiece simulation

Function

The actual part program is completely calculated in the tool simulation and the result is graphically displayed in the user interface. The result of programming is verified without traversing the machine axes. Incorrectly programmed machining steps are detected at an early stage and incorrect machining on the workpiece prevented.

Simulation NCK

The simulation uses its own NCK instance (simulation NCK). Therefore, before a simulation is started, the real NCK must be aligned to the simulation NCK. With this alignment, all active machine data are read out of the NCK and read into the simulation NCK. The NCK and cycle machine data are included in the active machine data.

Compile cycles in simulation (only 840D sl)

Up to SW 4.4, no compile cycles are supported, from SW 4.4 and higher only selected compile cycles (CC) are supported for the workpiece simulation. The machine data of the supported compile cycles are aligned once after the control has powered-up. An alignment with "simulation start" does not take place!

Note

In part programs, CC-specific language commands and machine data of unsupported CCs cannot be used (see also paragraph "CC-commands in the part program"). Special motion of supported CCs (OEM transformations) are - under certain circumstances - incorrectly displayed.

CC-commands in the part program

Language commands in the part program of compile cycles that are not supported (OMA1 ... OMA5, OEMIPO1/2, G810 ... G829, own procedures and functions) therefore result in an alarm message and cancellation of the simulation without any individual handling.

Solution: Individually handle the missing CC-specific language elements in the part program ($P_SIM query). Example:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1 G01 X200 F500</td>
<td></td>
</tr>
<tr>
<td>IF (1==P_SIM)</td>
<td></td>
</tr>
<tr>
<td>N5 X300</td>
<td>CC not active for simulation.</td>
</tr>
</tbody>
</table>
9.7 Block search, types 1, 2, and 4:

Function

Block search offers the possibility of starting part program execution from almost any part program block.

This involves the NC rapidly performing an internal run through the part program (without traversing motions) to the selected target block during block search. Here, every effort is made to achieve the exact same control status as would result at the target block during normal part program execution (e.g. with respect to axis positions, spindle speeds, loaded tools, NC/PLC interface signals, variable values) in order to be able to resume automatic part program execution from the target block with minimum manual intervention.

Block search types

- **Type 1: Block search without calculation**
  Block search without calculation is used to find a part program block in the quickest possible way. No calculation of any type is performed. The control status at the target block remains unchanged compared to the status before the start of the block search.

- **Type 2: Block search with calculation at contour**
  Block search with calculation at contour is used to enable the programmed contour to be approached in any situation. On NC Start, the start position of the target block or the end position of the block before the target block is approached. This is traversed up to the end position. Processing is true to contour.

- **Type 4: Block search with calculation at block end point**
  Block search with calculation at block end point is used to enable a target position (e.g. tool change position) to be approached in any situation. The end position of the target block or the next programmed position is approached using the type of interpolation valid in the target block. This is not true to contour.

  Only the axes programmed in the target block are moved. If necessary, a collision-free initial situation must be created manually on the machine in “JOG REPOS” mode before the start of further automatic part program execution.

- **Type 5: Block search with calculation in “Program test” (SERUPRO) mode**
  SERUPRO (search run by programtest) is a cross-channel block search with calculation. Here, the NC starts the selected part program in “Program test” mode. On reaching the target block, the program test is automatically deselected. This type of block search also enables interactions between the channel in which the block search is being performed and synchronized actions as well as with other NC channels.

  See Section “Block search Type 5 (SERUPRO) (Page 493)."
Note
For further explanations regarding the block search, see Section "Behavior during block search" (Page 418).

Subsequent actions
After completion of a block search, the following subsequent actions may occur:

- **Type 1 - Type 5: Automatic Start of an ASUB**
  When the last action block is activated, a user program can be started as an ASUB.

- **Type 1 - Type 4: Cascaded block search**
  A further block search with a different target specification can be started from "Search target found".

9.7.1 Description of the function

Basic sequence for type 2 or type 4
1. User: Activation of the block search [type 2 or type 4](block search with calculation to ...) via the operator interface
2. Search for the target block with collection of auxiliary functions
3. Stop after "Search target found ⇒ display of alarm 10208 "Continue program with NC start"
4. User: NC start to execute the action block s⇒ DB21, ... DBX7.1 = 1 (NC start)
5. Execution of the action blocks
6. Last action block is activated ⇒ Automatic start of /*_N_CMA_DIR/_N_PROG_EVENT_SPF*/ (default) as an ASUB.
7. Last ASUP block ([REPOSA](#)) is activated ⇒ DB21, ... DBX32.6 = 1 (last action block active)
8. Optional: Execution of user-specific requirements via PLC user program
9. Display of the alarm 10208 "Continue program with NC start"?
10. User: Continue program with NC start ⇒ DB21, ... DBX7.1 = 1 (NC start)

Search target not found
If the search target is not found, alarm 15370 "Search target not found during block search" is displayed and the block search is canceled.
**Time sequence**

<table>
<thead>
<tr>
<th>Block search starting</th>
<th>Search target 1 found</th>
<th>Block search starting</th>
<th>Search target 2 found</th>
<th>NC Start action blocks being output</th>
<th>Last action block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block search active (DB21, ... DBX33.4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action block active (DB21, ... DBX32.3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last action block active (DB21, ... DBX32.6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel status Reset (DB21, ... DBX35.7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel status interrupted (DB21, ... DBX35.6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Action blocks**

During a block search type 2 or type 4 (block search with calculation to ...), actions such as tool (T, D), spindle (S), feedrate programming or M function outputs are collected. With NC start to execute the action blocks, the collected actions are output to the PLC.

**Note**

With NC start for the action blocks, the spindle programming collected during block search type 2 or type 4 (block search with calculation to ...) (S value, M3/M4/M5/M19, SPOS) becomes active.

The user has to ensure in the PLC user program that the tool can be operated or the spindle programming is reset or not output:

DB31, ... DBX2.2 = 1 (spindle reset).

**Supplementary conditions**

**Continuation mode after block search type 4**

If initial programming of an axis is performed incrementally to a block search type 4 (block search with calculation to block end point), the programmed incremental value can be added to the position value collected up to the search target or to the current actual value of the axis. The setting is made via:

SD42444 $SC\_TARGET\_BLOCK\_INCR\_PROG

The setting data is evaluated with NC start for outputting the action blocks.
9.7 Block search, types 1, 2, and 4:

**Single block**

If you do not wish to stop after each action block after the search target during block search type 2 or type 4 (block search with calculation to ...) is found and while function "Single block" (DB21, ... DBX0.4 == 1 (activate single block)) is active, this behavior can be deactivated in machine data:

MD10702 $MN_IGNORE_SINGLEBLOCK_MASK, Bit 3 = 1 (ignore signal block for action blocks)

**Interface signal "start block active"**

The interface signal is only set during block search type 2 (block search with calculation to contour):

- DB21, ... DBX32.4 = 1 (approach block active)

For block search type 4 (block search with calculation to block end point), the interface signal is not set because no approach block is generated here (approach block equal to target block).

**Type of interpolation of the target block**

For a block search type 4 (block search with calculation to block end point), the start movement is performed in the interpolation type valid in the target block. For interpolation types other than linear interpolation (G0 or G1), the start movement can be canceled with an alarm (e.g. circle end point error in circular interpolation G2 or G3).

---

9.7.2 Block search in connection with other NCK functions

9.7.2.1 ASUB after and during block search

**Block search type 2 and type 4: Synchronization of the channel axes**

If an ASUP is started after a block search type 2 or type 4 (block search with calculation to ...), the actual positions of all channel axes are synchronized.

**Effect**

If the following system variables are read in the ASUP, they contain the following values:

- $P_EP: Current actual position of the channel axis in the WCS
- $AC_RETPOINT: Collected block search position of the channel axis in the WCS

**Block search type 2: ASUP completion**

For block search type 2 (block search with calculation to contour), the following command REPOSA (reapproach to the contour; linear; all channel axes) must be programmed by completion of the ASUP.
Effect
• All channel axes are moved to their search position that was collected during the block search.
• $P_{EP} == "$accumulated search position of the channel axis (WCS)"

Block search type 4: REPOS behavior
After block search type 4 (block search with calculation to block end point), no automatic repositioning is triggered during the period described by the beginning and end by the REPOS command:
• Begin: NC/PLC interface signal DB21, … DBX32.6 = 1 (last action block active)
• End: Continuing program execution with NC start.

The start point of the approach movement are the current positions of the channel axes at the time of the NC start command. The end point results from the other transversing movements programmed in the part program.

For block search type 4, no approach movement is generated by the NC.

Effect:
• After exiting the ASUP, the system variable $P_{EP} (programmed end position) contains the actual position, at which the channel axes were positioned by the ASUP or manually (mode: JOG).
$P_{EP} == "$current actual position of the channel axis in the WCS"

9.7.2.2 PLC actions after block search
If all action blocks have been executed by the NC and actions by the PLC, e.g. starting an ASUP to perform a tool change, or of the operator, e.g. restore, are possible, the following channel-specific NC/PLC interface signal is set:
• DB21, … DB32.6 = 1 (last action block active)

Parameterizing the time of alarm output
To inform the operator that NC start is required in the channel to continue program execution, alarm 10208 "Continue program with NC start" is displayed.
When the alarm is displayed can be set via machine data:
MD11450 $MN_SEARCH_RUN_MODE, Bit 0 = <value>

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0       | With the change of the last action block after a block search, the following takes place:  
  ● Execution of the part program is stopped  
  ● DB21, ... DBB32.6 = 1 (last action block active)  
  ● Display alarm 10208 |
| 1       | With the change of the last action block after a block search, the following takes place:  
  ● Execution of the part program is stopped  
  ● DB21, ... DBB32.6 = 1 (last action block active)  
  ● Only display alarm 10208 if DB21, ... DBX1.6 == 1 (PLC action ended) |

In combination with the alarm, the following interface signals are set:
- DB21, ... DBX36.7 = 1 (NC alarm with machining standstill is pending)
- DB21, ... DBX36.6 = 1 (NC alarm channel-specific is pending)

9.7.2.3 Spindle functions after block search

Control system response and output

Whether the spindle-specific auxiliary functions collected during block search are automatically output to the PLC in the action block or user-specifically at a later time can be set in
MD11450 $MN_SEARCH_RUN_MODE, Bit 2 = <value>

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output of spindle-specific auxiliary functions collected during block search (M3, M4, M5, M19, M70) in action blocks.</td>
</tr>
</tbody>
</table>
| 1       | The spindle-specific auxiliary functions collected during block search are not output in the action block.  
  They can be output at a later time, for example, in an ASUP. The spindle-specific auxiliary functions collected are stored in system variables: |

System variables

The spindle-specific auxiliary functions collected during block search are stored in the following system variables:

<table>
<thead>
<tr>
<th>System variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_SEARCH_S [ &lt;n&gt; ]</td>
<td>The spindle speed or cutting speed last programmed</td>
</tr>
<tr>
<td>$P_SEARCH_SDIR [ &lt;n&gt; ]</td>
<td>Spindle direction last programmed</td>
</tr>
<tr>
<td>$P_SEARCH_SGEAR [ &lt;n&gt; ]</td>
<td>Gear stages last programmed M function</td>
</tr>
<tr>
<td>$P_SEARCH_SMODE [ &lt;n&gt; ]</td>
<td>Spindle mode last programmed</td>
</tr>
<tr>
<td>$P_SEARCH_SPOS [ &lt;n&gt; ]</td>
<td>Spindle position or traverse path last programmed by means of M19, SPOS or SPOSA</td>
</tr>
</tbody>
</table>
### 9.7.2.4 Reading system variables for a block search

In part programs, values can be read from the areas processing, main run, or servo/drive via system variables:

<table>
<thead>
<tr>
<th>System variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{...}$</td>
<td>Preprocessing-related system variables contain programmed values</td>
</tr>
<tr>
<td>$A_{...}$</td>
<td>Main-run-related system variables contain current values</td>
</tr>
<tr>
<td>$V_{...}$</td>
<td>Servo/drive-related system variables contain current values</td>
</tr>
</tbody>
</table>

Because no blocks enter the main run during a block search of type 2 and type 4 (block search with calculation to ...), main run and servo/drive-related system variables are not changed during the block search. Where necessary, for these system variables, the block search must specially handled in the NC program by querying whether a block search is active $P_{SEARCH}$ (block search active).

Preprocessing-related system variables provide correct values in all search types.

### 9.7.3 Automatic start of an ASUB after a block search

#### Parameter assignment

**Making the function effective**

The automatic ASUB start after a block search is activated by the following MD setting:

MD11450 $MN_{SEARCH\_RUN\_MODE}$, bit 1 = 1

**Program to be activated**

In the default setting, the program _N_PROG_EVENT_SPF_ is activated from the directory _N_CMA_DIR_ as ASUB after the block search by changing the last action block. If another program is to be activated, then the name of this user program must be entered in the following machine data:
MD11620 $MN_PROG_EVENT_NAME

Behavior when the single-block processing is set
The following channel-specific machine data are used to set whether the activated ASUP will be processed without interruption although single-block processing is set or whether single-block processing will be activated:

MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK, bit 4 = <value>

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Single-block processing is active.</td>
</tr>
<tr>
<td>1</td>
<td>Single-block processing is suppressed.</td>
</tr>
</tbody>
</table>

Behavior when the read-in disable is set
The channel-specific machine data is used to set whether the ASUP will be processed without interruption although read-in disable (DB21, ... DBX6.1 = 1) is set, or whether the read-in disable will be effective:

MD20107 $MC_PROG_EVENT_IGN_INHIBIT, Bit 4 = <value>

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Read-in disable is active.</td>
</tr>
<tr>
<td>1</td>
<td>Read-in disable is suppressed.</td>
</tr>
</tbody>
</table>

Note
For more information on parameterizing MD11620, MD20108, and MD20107, see Section "Parameterization (Page 549)."

Programming
The event by which the ASUP has been started is stored in the system variable $P_PROG_EVENT. On automatic activation after a block search, $P_PROG_EVENT returns the value "5."

Sequence
Sequence of automatic start of an ASUB after a block search
1. User: Activation of the block search **type 2** or **type 4** (block search **with calculation** to ...) via the operator interface
2. Search for the target block with collection of auxiliary functions
3. Stop after "Search target found => display of alarm 10208 "Continue program with NC start"
4. User: NC start to execute the action block s => DB21, ... DBX7.1 = 1 (NC start)
5. Execution of the action blocks
6. Last action block is activated => Automatic start of \_/N_CMA_DIR/_N_PROG_EVENT_SPF (default) as an ASUB.
7. Last ASUP block (REPOSA) is activated ⇒ DB21, ... DBX32.6 = 1 (last action block active)
8. Optional: Execution of user-specific requirements via PLC user program
9. Display of the alarm 10208 "Continue program with NC start"?

   **Note**
   With MD11450 $MN_SEARCH_RUN_MODE, Bit 0 == 1, alarm 10208 will only be output after enabling by the PLC user program (DB21, ... DBX1.6 = 1 (PLC action ended)).

10. User: Continue program with NC start ⇒ DB21, ... DBX7.1 = 1 (NC start)

### 9.7.4 Cascaded block search

**Functionality**

The "Cascaded block search" function can be used to start another block search from the status "Search target found". The cascading can be continued after each located search target as often as you want and is applicable to the following block search functions:

- Type 1 block search without calculation
- Type 2 block search with calculation at contour
- Type 3 block search with calculation at block end point

**Note**

Another "cascaded block search" can be started from the stopped program execution only if the search target has been found.

**Activation**

The "cascaded block search" is configured in the existing machine data:

MD11450 $MN_SEARCH_RUN_MODE

- Cascaded block search is enabled (i.e., several search targets can be specified) with Bit 3 = 0 (FALSE).
- For compatibility reasons, the cascaded block search can be disabled with Bit 3 = 1 (TRUE). By default, the cascaded block search is set with Bit 3 = 0.

**Execution behavior**

**Search target found, restart search**

When the search target is reached, the program execution stops and the search target is displayed as a current block. After each located search target, a new block search can be repeated as often as you want.

**Change search target specifications**
You can change the search target specifications and block search function prior to each block search start.

**Example: Sequence with cascaded block search**

- **RESET**
- Block search up to search target 1
- Block search up to search target 2 → "Cascaded block search"
- NC Start for output of the action blocks → Alarm 10208
- NC Start → Continue program execution

![Chronological order of interface signals](image)

**9.7.5 Examples for block search with calculation**

**Selection**

From the following examples, select the type of block search that corresponds to your task.
Type 4 block search with calculation at block end point

Example with automatic tool change after block search with active tool management:

1. Set machine data:
   MD11450 $MN_ SEARCH_RUN_MODE to 1
   MD11602 $MN_ASUB_START_MASK Bit 0 = 1 (ASUB Start from stopped state)

2. Select ASUB "BLOCK_SEARCH_END" from PLC via FB4 (see also Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 821)").

3. Load and select part program "WORKPIECE_1".

4. Search to block end point, block number N220.

5. HMI signals "Search target found".

6. NC Start for output of action blocks.

7. With PLC signal:
   DB21... DB32.6 (last action block active)
   the PLC starts ASUB "BLOCK_SEARCH_END" via FC9 (see also Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 821)").

8. After the end of the ASUB (can be evaluated, e.g. via M function M90 to be defined, see example for block N1110), the PLC sets signal
   DB21, ... DBX1.6 (PLC action complete).
   Alternatively, NC/PLC interface signal:
   DB21-DB30 DBB318 bit 0 (ASUB is stopped)
   can be scanned.
   As a result, Alarm 10208 is displayed, i.e. other actions can now be performed by the operator.

10. Continue part program with NC Start.

Figure 9-3  Approach motion for search to block end point (target block \textit{N220})

\textbf{Note}

"Search to contour" with target block \textit{N220} would generate an approach motion to the tool change point (start point of the target block).

\textbf{Type 2 block search with calculation at contour}

Example with automatic tool change after block search with active tool management:

1. to 3. Same as example for Type 4 block search

4. Search to contour, block number \textit{N260}

5. to 10. Same as example for Type 4 block search
9.7 Block search, types 1, 2, and 4:

Figure 9-4  Approach motion for search to contour (target block N260)

Note
"Search to block end point" with target block N260 would result in Alarm 14040 (circle end point error).

Part programs for Type 4 and Type 2

**PROC WORKPIECE_1**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>; Main program</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>;Machine contour section 1 with &quot;CUTTER_1&quot; tool</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N100 G0 G40 X200 Y200</td>
<td>; Deselect radius compensation</td>
</tr>
<tr>
<td>N110 Z100 D0</td>
<td>; Deselect length correction</td>
</tr>
<tr>
<td>;End of contour section 1</td>
<td></td>
</tr>
<tr>
<td>;Machine contour section 2 with &quot;CUTTER_2&quot; tool</td>
<td></td>
</tr>
<tr>
<td>N200 T=&quot;CUTTER_2&quot;</td>
<td>; Preselect tool</td>
</tr>
<tr>
<td>N210 WZW</td>
<td>; Call tool change routine</td>
</tr>
<tr>
<td>N220 G0 X170 Y30 Z10 S3000 M3 D1</td>
<td>; Approach block for contour section 2</td>
</tr>
<tr>
<td>N230 Z-5</td>
<td>; Infeed</td>
</tr>
<tr>
<td>N240 G1 G64 G42 F500 X150 Y50</td>
<td>; Start point of contour</td>
</tr>
<tr>
<td>N250 Y150</td>
<td></td>
</tr>
<tr>
<td>N260 G2 J50 X100 Y200</td>
<td></td>
</tr>
<tr>
<td>N270 G1 X50</td>
<td></td>
</tr>
</tbody>
</table>
### Program code

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N280 Y50</td>
<td></td>
</tr>
<tr>
<td>N290 X150</td>
<td></td>
</tr>
<tr>
<td>N300 G0 G40 G60 X170 Y30</td>
<td>; Deselect radius compensation</td>
</tr>
<tr>
<td>N310 Z100 D0</td>
<td>; Deselect length correction</td>
</tr>
<tr>
<td></td>
<td>End of contour section 2</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>M30</td>
<td></td>
</tr>
<tr>
<td>PROC WZW</td>
<td></td>
</tr>
</tbody>
</table>

:Tool change routine

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N500 DEF INT TNR_AKTIV</td>
<td>; Variable for active T number</td>
</tr>
<tr>
<td>N510 DEF INT TNR_VORWAHL</td>
<td>; Variable for preselected T number</td>
</tr>
<tr>
<td>N520 TNR_AKTIV = $TC_MPP6[9998,1]</td>
<td>; Read T number of active tool</td>
</tr>
<tr>
<td>N530 GETSELT(TNR_VORWAHL)</td>
<td>; Read T number of preselected tool</td>
</tr>
<tr>
<td>; Execute tool change only if tool is not yet active</td>
<td></td>
</tr>
<tr>
<td>N540 IF TNR_AKTIV == TNR_VORWAHL GOTOF ENDE</td>
<td></td>
</tr>
<tr>
<td>N550 G0 G40 G60 G90 SUPA X450 Y300 Z300 D0</td>
<td>; Approach tool change position</td>
</tr>
<tr>
<td>N560 M6</td>
<td>; Execute tool change</td>
</tr>
<tr>
<td>; END: M17</td>
<td></td>
</tr>
<tr>
<td>PROC SUCHLAUF_ENDE SAVE</td>
<td></td>
</tr>
</tbody>
</table>

:ASUB for calling the tool change routine after block search

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1000 DEF INT TNR_AKTIV</td>
<td>; Variable for active T number</td>
</tr>
<tr>
<td>N1010 DEF INT TNR_VORWAHL</td>
<td>; Variable for preselected T number</td>
</tr>
<tr>
<td>N1020 DEF INT TNR_SUCHLAUF</td>
<td>; Variable for T number determined in search</td>
</tr>
<tr>
<td>N1030 TNR_AKTIV = $TC_MPP6[9998,1]</td>
<td>; Read T number of active tool</td>
</tr>
<tr>
<td>N1040 TNR_SUCHLAUF = $P_TOOLNO</td>
<td>; Read T number determined by search</td>
</tr>
<tr>
<td>N1050 GETSELT(TNR_VORWAHL)</td>
<td>; Read T number of preselected tool</td>
</tr>
<tr>
<td>N1060 IF TNR_AKTIV == TNR_SUCHLAUF GOTOF ASUP_ENDE</td>
<td></td>
</tr>
<tr>
<td>N1070 T = $TC_TP2[TNR_SUCHLAUF]</td>
<td>; T selection by tool name</td>
</tr>
<tr>
<td>N1080 WZW</td>
<td>; Call tool change routine</td>
</tr>
<tr>
<td>N1090 IF TNR_VORWAHL == TNR_SUCHLAUF GOTOF ASUP_ENDE</td>
<td></td>
</tr>
<tr>
<td>N1100 T = $TC_TP2[TNR_VORWAHL]</td>
<td>; Restore T preselection by tool name</td>
</tr>
<tr>
<td>ASUP_ENDE:</td>
<td></td>
</tr>
<tr>
<td>N1110 M90</td>
<td>; Feedback to PLC</td>
</tr>
<tr>
<td>N1120 REPOSA</td>
<td>; ASUB end</td>
</tr>
</tbody>
</table>
9.7.6 Supplementary conditions

9.7.6.1 Compressor functions (COMPON, COMPCURV, COMPCAD)

- If the target block for block search type 2 or type 4 (block search with calculation to ...) is in a program section in which a compressor function (COMPON, COMPCURV, COMPCAD) is active, positions are approached on the path calculated by the compressor on repositioning. These positions must precisely match the positions on the path programmed in the part program.

- If programmed blocks are eliminated from the part program during compression, these blocks will not be found at the target block in the block search ⇒ Alarm 15370 "Search target not found."

9.8 Block search Type 5 (SERUPRO)

9.8.1 Description of the function

Block search type 5, block search with calculation in the "Program test" mode (SERUPRO, "Search-Run by Program test") enables a cross-channel block search with calculation at a selectable interruption point. Taking into account existing program coordination commands, all the status data required to continue the program in the interrupted channels is determined during SERUPRO and then the NC and PLC set to a state permitting the program continuation.

Before repositioning with subsequent continuation of the program execution, all the output states that may still be required can be automatically generated via a user-specific ASUB.

Channels

In combination with the HMI, SERUPRO is provided for the following channels:

- For the current SERUPRO channel only (1)
- For all channels with the same workpiece name as the SERUPRO channel (2)
- For all channels with the same mode group as the SERUPRO channel (3)
- For all channels of the NCU (4)

The scope of channels for SERUPRO is selected by means of configuration file maschine.ini, in Section [BlockSearch]:

<table>
<thead>
<tr>
<th>Section [BlockSearch]</th>
<th>Enable search function for HMI and select search configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeruproEnabled=1</td>
<td>;SERUPRO softkey available for HMI. Default value is (1)</td>
</tr>
<tr>
<td>SeruproConfig=1</td>
<td>;Number (1) to (4) of above indicated channel grouping. Default value is (1)</td>
</tr>
</tbody>
</table>

All other channels started with SERUPRO are operated in "Self-Acting SERUPRO" mode. Only the channel in which a target block has been selected can be started with a block search in SERUPRO mode.
SERUPRO is activated via the HMI. SERUPRO is operated using the "Prog.Test Contour" softkey.

SERUPRO uses REPOS to approach the target block.

Chronological sequence of SERUPRO

1. Via HMI, softkey "Pog. test contour" and the search target are operated.
2. The NC now automatically starts the selected program in "Program test" mode.
   - In this mode, axes are not traversed.
   - Auxiliary functions $A_{OUT}$ and the direct PLC IO are output.
   - The auxiliary functions of the target block are not output.
3. The NC stops at the beginning of the target block, deselects the program test internally, and displays the Stop condition "Wait: Search target found".
4. If the user-specific ASUB "PROGEVENT.SP" is available, it is started automatically.
5. Repositioning is performed with the next NC start (REPOS).
   The REPOS operation is performed via a system ASUB and can be extended using the "Editable ASUB" function.

Boundary conditions for block search SERUPRO

The SERUPRO function may only be activated in "AUTOMATIC" mode and may only be aborted in program state (channel state RESET).

If in normal mode only the PLC starts commonly several channels, then this can be simulated by SERUPRO in each channel.

With machine data setting:
MD10708 $MN_{SERUPRO\_MASK}$, bit 1 = 0,
alarm 16942: "Channel %1 Start program command action %2<ALNX> not possible" aborts the simulation if part program command START is used.

Machine data:
MD10707 $MN_{PROG\_TEST\_MASK}$ allows shutdown in the stopped state and has no effect on the SERUPRO operation. The default setting allows program testing to be deactivated only in the RESET state.

Note

After program testing has been deactivated, a REPOS operation is initiated that is subject to the same restrictions as a SERUPRO approach operation. Any adverse effects can be inhibited using an ASUB.
Controlling SERUPRO behavior

For the functions listed below as an example, the SERUPRO behavior can be set specifically for the NC:

- Programmed stop (M0)
- Program coordination command START
- Group SERUPRO
- Cross-channel exiting of SERUPRO
- Override

MD10708 $MN_SERUPRO_MASK = <behavior with SERUPRO>

Channel-specific initial settings for SERUPRO

The channel-specific basic settings are normally specified with the following machine data after a part program start:

MD20112 $MC_START_MODE_MASK= <initial settings>

You can specify your own initial settings for SERUPRO which replace the initial settings from MD20112:

MD22620 $MN_START_MODE_MASK_PRT = <SERUPRO initial settings>

The SERUPRO initial settings must be explicitly released via:

MD22621 $MC_ENABLE_START_MODE_MASK_PRT = 1

NC/PLC interface signal "Block search via program test is active"

The block search via program test is displayed using the NC/PLC interface signal:

DB21, ... DBX318.1 == 1

The interface signal is set from the start of the block search until the target block is inserted into the main run.

For user-defined ASUB after the SERUPRO operation

Note

If the machine manufacturer decides to start an ASUB after the SERUPRO operation as described in point 7, the following must be observed:

Stopped status acc. to point 6:

Machine data:

MD11602 $MN_ASUP_START_MASK

and

MD11604 $MN_ASUP_START_PRIO_LEVEL

allow the NCK to start the ASUB from stopped status automatically via the FC9 block.
Acknowledgement of FC9 only after completion of REPOS block:
The ASUB can only be signaled as complete from the FC9 block with "ASUB Done" if the REPOS block has also been completed.

Deselection of assigned REPOS operation after point 8:
The start of the ASUB deselects the assigned REPOS operation!
Therefore, the ASUB should be ended with REPOSA in order to retain the REPOS operation.

Deleting an unwanted REPOS operation:
The unwanted REPOS operation is deleted by completing the ASUB with M17 or RET.

Special handling of ASUB:
As a basic rule, an ASUB that ends with REPOS and is started from stopped state receives special treatment.
The ASUB stops automatically before the REPOS block and indicates this via:
DB21, ... DBX318.0 (ASUB is stopped)

Automatic ASUB start
The ASUB in path:
/ N_CMA_DIR/ N_PROG_EVENT_SPF
is started automatically in SERUPRO approach with machine data:
MD11450 $MN_SEARCH_RUN_MODE, bit 1 = 1
according to the following sequence:
1. The SERUPRO operation has been performed completely.
2. The user presses "NC start".
3. The ASUB is started.
4. The NC stops automatically before the REPOS part program command and the message "Press NC start to continue the program" appears.
5. The user presses "NC start" again.
6. The NC executes the repositioning motion and continues the part program at the target block.

Note
The automatic ASUB start with MD11450 requires Starts to continue the program.
The procedure is in this respect similar to other search types.
9.8.2 Repositioning to the contour (REPOS)

The "Reposition to the contour" (REPOS) function can be used to continue an interrupted machining at the interrupted location. Unlike REPOS, SERUPRO permits the "refetching" or "repetition" of a program section. For this purpose, once SERUPRO has found the target block, the contour is positioned at the location selected as REPOS mode and the machining continued.

SERUPRO: Set REPOS response

The REPOS behavior, i.e. the behavior in the repositioning block, is set with the following machine data:

\[
\text{MD11470 } $\text{MN\_REPOS\_MODE\_MASK} = <\text{REPOS mode}>
\]

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>An interrupted delay time is repeated</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>An interrupted delay time is continued</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>DB31, ..., DBX10.0 (REPOSDELAY) not considered in the repositioning block.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>DB31, ..., DBX10.0 (REPOSDELAY) considered in the repositioning block.</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>SERUPRO: Only path axes traverse in the repositioning block</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>SERUPRO: Path and positioning axes traverse concurrently in the repositioning block</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>REPOS: Only path axes traverse in the repositioning block</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>REPOS: Path and positioning axes traverse concurrently in the repositioning block</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>During the interruption, changed feedrates and spindle speeds act only after the first part program block following the interruption location</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>During the interruption, changed feedrates and spindle speeds act following the interruption location</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>SERUPRO: In the repositioning block, neutral axes and positioning spindles traverse as path axes.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>SERUPRO: In the repositioning block, neutral axes and positioning spindles traverse as command axes.</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>@@ @</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>@@ @</td>
</tr>
</tbody>
</table>

The level of interface signal: 

\[
\text{DB31, ..., DBX10.0 (REPOSDELAY)}
\]

is read if REPOSA is interpreted. Axes, which are neither geometry nor orientation axes, are then excluded by REPOS and are not moved.

⚠️ CAUTION

Risk of collision

\[
\text{MD11470 } $\text{MN\_REPOS\_MODE\_MASK}$, Bit 3 or Bit 4 = 1
\]

The user alone is responsible for ensuring that the concurrent traversal of the axes in the repositioning block does not cause any collision on the machine.
Repositioning with controlled REPOS

The REPOS mode can be specified for the path axes via the NC/PLC interface:

DB21, ..., DBX31.0 - .2 (REPOS mode A, B, C)

This mode is programmed in the part program and defines the approach behavior (see Section "Repositioning with controlled REPOS (Page 505)").

The REPOS behavior of individual axes can also be controlled via NC/PLC interface signals and is enabled with machine data:

MD11470 $MN_REPOS_MODE_MASK BIT 2==1.

Path axes cannot be influenced individually. For all other axes that are not geometry axes, REPOS of individual axes can be prevented temporarily and also delayed. NC/PLC interface signals can be used to subsequently re-enable or to continue blocking individual channel axes that REPOS would like to traverse.

⚠️ DANGER

Risk of collision

Signal:

DB31, ..., DBX2.2 (delete distance-to-go, axis-specific) produces the following dangerous behavior with "Prevent repositioning of individual axes" via:

MD11470 $MN_REPOS_MODE_MASK (Bit 2==1).

As long as an axis is programmed incrementally after the interruption, the NC approaches different positions than those approached with no interruption (see example below).

Example: Axis is programmed incrementally

Axis A is positioned at 11° before the REPOS operation; the programmed operation in the interruption block (target block for SERUPRO) specifies 27°.

Any number of blocks later, this axis is programmed to move incrementally through 5° with:


With interface signal:

DB31, ..., DBX10.0 (REPOSDELAY)

the axis does not traverse in the REPOS operation and is moved to 32° with N1010.

(The user may have to deliberately acknowledge the travel from 11° to 27°.)

Caution:
The axis is programmed incrementally after the interruption.

In the example, the NC moves to 16° (instead of 32°).

Starting axes individually

The REPOS behavior for SERUPRO approach with several axes is selected with:

MD11470 $MN_REPOS_MODE_MASK BIT 3 == 1

The NC commences SERUPRO approach with a block that moves all positioning axes to the programmed end and the path axis to the target block.
The user starts the individual axes by selecting the appropriate feedrate enables. The target block motion is then executed.

**Repositioning positioning axes in the repositioning block**

Positioning axes are not repositioned in the residual block but rather in the repositioning block, and their effect is not limited to the block search via program test on SERUPRO approach.

<table>
<thead>
<tr>
<th>MD11470 $MN_REPOS_MODE_MASK</th>
<th>Bit 3=1</th>
<th>for block search via program test (SERUPRO)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bit 4=1</td>
<td>for each REPOS</td>
</tr>
</tbody>
</table>

**Note**

If neither bit 3 nor bit 4 is set, "non-path axes" are repositioned in the residual block in this phase.

**Prefer or ignore REPOS**

Further REPOS adaptations can be made by setting the bits in:

MD11470 $MN_REPOS_MODE_MASK

- **Bit 5=1** Modified feedrates and spindle speeds are valid immediately in the residual block and are given priority. This behavior relates to every REPOS operation.
- **Bit 6=1** Neutral axes and positioning spindles are repositioned after SERUPRO. Neutral axes that are not allowed to be further repositioned must receive interface signal:
  DB31, ... DBX10.0 (REPOSDELAY)
  This cancels the REPOS motion.
- **Bit 7=1** The level of interface signal:
  DB31, ... DBX10.0 (REPOSDELAY)
  is read if REPOSA is interpreted.
  Axes, which are neither geometry nor orientation axes, are then excluded by REPOS and are not moved.
  Remark: REPOSDELAY is changed from edge to level evaluation.

**Delayed approach of axis with REPOS offset**

With the axial level-triggered NC/PLC interface signal axis/spindle (PLC→NCK):

DB31, ... DBX10.0 (REPOSDELAY)

the REPOS offset for this axis is traversed only after the next time it is programmed with the edge of IS:

DB21, ... DBX31.4 (REPOSMODEEDGE).
Whether this axis is currently subject to a REPOS offset can be scanned via synchronized actions with $AA_REPOS_DELAY.

<table>
<thead>
<tr>
<th>CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk of collision</td>
</tr>
<tr>
<td>Interface signal:</td>
</tr>
<tr>
<td>DB31, ... DBX10.0 (REPOSDELAY)</td>
</tr>
<tr>
<td>has no effect on machine axes that form a path.</td>
</tr>
</tbody>
</table>

Whether an axis is a path axis can be determined with:
DB31, ... DBX76.4 (path axis).

Acceptance timing of REPOS NC/PLC interface signals

With the 0/1 edge of the channel-specific NC/PLC interface signal (PLC→NCK):
DB21, ... DBX31.4 (REPOSMODEEDGE)

the level signals of:
DB21, ... DBX31.0-31.2 (REPOSPATHMODE0 to 2)

and
DB31, ... DBX10.0 (REPOSDELAY)

are transferred to the NC.

The levels relate to the current block in the main run. There are two different cases:

1. **One** repositioning block of a currently active REPOS operation is contained in the main run.
   The active REPOS operation is aborted, restarted and the REPOS offsets controlled via the signals:
   DB21, ... DBX31.0-31.2 (REPOSPATHMODE0 to 2)
   and
   DB31, ... DBX10.0 (REPOSDELAY).

2. No repositioning block of a currently active REPOS operation is contained in the main run.
   Each future REPOS operation wishing to reapproach the current main program block is controlled by the level of interface signal:
   DB21, ... DBX31.0-31.2 (REPOSPATHMODE0 to 2)
   and
   DB31, ... DBX10.0 (REPOSDELAY).
Note

In the current ASUB, DB21, ... DBX31.4 (REPOSMODEEDGE) does not affect the final REPOS unless this signal applies to the REPOS blocks.

In case 1, the signal is allowed only in the stopped state.

Response to RESET:

● NCK has acknowledged the PLC signal
  DB21, ... DBX31.4 (REPOSMODEEDGE) == 1 AND
  DB21, ... DBX319.0 (REPOSMODEEDGEACKN) == 1
  If a RESET occurs in this situation, DB21, ... DBX319.1 - .3 (Repos Path Mode Ackn) is deleted.

● NCK has not yet acknowledged the PLC signal:
  DB21, ... DBX31. (REPOSMODEEDGE 4) == 1 AND
  DB21, ... DBX319.0 (REPOSMODEEDGEACKN) == 0
  If a RESET occurs in this situation, then:
  DB21, ... DBX319.0 (REPOSMODEEDGEACKN) = 0
  DB21, ... DBX319.1–319.3 (Repos Path Mode Ackn) = 0

Controlling SERUPRO approach with NC/PLC interface signals

The SERUPRO approach can be used with:
DB21, ... DBX31.4 (REPOSMODEEDGE)
and the associated signals in the following phases:

● Between "Search target found" and "Start SERUPRO ASUB"
● From "SERUPO-ASUB stops automatically before REPOS" to "Target block is executed"

While the SERUPRO ASUB is being executed, e.g. in the program section before the REPOS operation, the interface signal does not affect the SERUPRO positioning.

REPOS operations with NC/PLC interface signals

Control REPOS with NC/PLC interface signals

REPOS offsets can be positively influenced with the following channel-specific NC/PLC interface signals from the PLC:

● DB21, ... DBX31.0-31.2 (REPOSPATHMODE0 to 2) channel-specific *
● DB21, ... DBX31.4 (REPOSMODEEDGE) channel-specific
● DB31, ... DBX10.0 (REPOSDELAY) *axis/spindle (This axial interface does not affect machine axes that form a path.)
● DB31, ... DBX72.0 (REPOSDELAY) axis/spindle

* These signals are available in the respective DB of the HMI or PLC.
REPOS acknowledgement signals

The following NC/PLC interface signals can be used to acknowledge from the NC, functions that control the REPOS response via PLC:

- DB21, ... DBX319.0 (REPOSMODEEDGEACKN) channel-specific
- DB21, ... DBX319.1-319.3 (Repos Path Mode Ackn0 to 2) channel-specific
- DB21, ... DBX319.5 (Repos DEFERRAL Chan) channel-specific
- DB31, ... DBX70.0 (Repos offset) axis/spindle
- DB31, ... DBX70.1 (Repos offset valid) axis/spindle
- DB31, ... DBX70.2 (Repos Delay Ackn) axis/spindle
- DB31, ... DBX76.4 (path axis) axis/spindle

For further information, see "REPOS offset in the interface"

REPOS acknowledgement operations

With the channel-specific NC/PLC interface signal:
DB21, ... DBX319.0 (REPOSMODEEDGEACKN)

a "handshake" is established by the interface signal:
DB21, ... DBX31.4 (REPOSMODEEDGE)

which is recognized by the NC and acknowledged with DB21, ... DBX319.0 to the PLC.

---

Note

If the NC has not yet acknowledged interface signal:
DB21, ... DBX31.4 (REPOSMODEEDGE)
with interface signal:
DB21, ... DBX319.0 (REPOSMODEEDGE)
a RESET in this situation causes the program to abort, and the REPOS that is to be used to control the REPOSPATHMODE is not executed.

A REPOSMODE issued from the PLC is acknowledged by the NC using the following interface signals:

- DB21, ... DBX319.1-319.3 (Repos Path Mode Ackn)
- DB31, ... DBX10.0 (Repos Delay)
- DB31, ... DBX70.2 (Repos Delay Ackn)

Example

A part program is stopped at \texttt{N20} (→ time (2) in figure). The NCK stops according to the braking ramp. After the PLC has specified the REPOSPATHMODE, the NCK accepts the REPOSPATHMODE with the 0/1 edge of REPOSMODEEDGE at → Time (3). Repos Path Mode Ackn remains set until the ASUB is initiated (→ Time (4)). The REPOS command is started in the ASUB (→ Time (5)). The ASUB RESET block is activated again (→ Time (6)):
NC sets acknowledgement again

Phase with REPOSPATHMODE still active (residual block of the program stopped at \(\rightarrow\) Time (2) is not yet completely executed).

As soon as the REPOS repositioning motion of the ASUB is executed, the NCK sets the "Repos Path Mode Ackn" again (\(\rightarrow\) Time (5)). If no REPOSPATHMODE has been preselected via an NC/PLC interface signal, the programmed REPOS mode is displayed.

"Repos Path Mode Ackn" is canceled when the residual block is activated (\(\rightarrow\) Time (6)). The part program block N30 following the block at \(\rightarrow\) Time (2) is resumed.

Interface signal:
DB31, ... DBX70.2 (Repos Delay Ackn) is defined in the same way.

DB31, ... DBX70.1 (Repos offset valid) = 1, if:
DB21, ... DBX319.1-319.3 (Repos Path Mode Ackn0 to 2) = 4 (RMNBL).

Valid REPOS offset

At the end of the SERUPRO operation, the user can read out the REPOS offset via the axis/spindle NC/PLC interface signal (NCK→PLC):
DB31, ... DBX70.0 (REPOS offset).
The effects of this signal on the relevant axis are as follows:

Value 0: No REPOS offset is applied.
Value 1: REPOS offset is applied.

Range of validity

Interface signal:
DB31, ... DBX70.0 (REPOS offset)
is supplied at the end of the SERUPRO operation.
The REPOS offset is invalidated at the start of a SERUPRO ASUB or the automatic ASUB start.

Updating the REPOS offset within the scope

Between the SERUPRO end and SERUPRO start, the axis can be moved in JOG mode with a mode change.
In JOG mode, the user manually moves the axis over the REPOS offset path in order to set interface signal:
DB31, ... DBX70.0 (REPOS offset) to the value 0.
Within the range of validity, the axis can also be traversed using FC18, whereby the IS DB31, ... DBX70.0 (REPOS offset) is continuously updated.

Displaying the range of validity

The range of validity of the REPOS offset is indicated with interface signal:
DB31, ... DBX70.1 (REPOS offset valid)
It is indicated whether the REPOS offset calculation was valid or invalid:

Value 0: The REPOS offset of this axis is calculated correctly.
Value 1: The REPOS offset of this axis cannot be calculated, as the REPOS has not yet occurred, e.g. it is at the end of the ASUB, or no REPOS is active.

REPOS offset after an axis interchange

The group signal:
DB21, ... DBX319.5 (Repos DEFERRAL Chan)
can be used to determine whether a valid REPOS offset has taken place:

Value 0: All axes currently controlled by this channel have either no REPOS offset or their REPOS offsets are invalid.
Value 1: Miscellaneous.
REPOS offset with synchronized synchronous spindle coupling

When repositioning with SERUPRO, processing continues at the point of interruption. If a synchronous spindle coupling was already synchronized, there is no REPOS offset of the following spindle and no synchronization path is present. The synchronization signals remain set.

Search target found on block change

The axial NC/PLC interface signal:
DB31, ... DBX76.4 (path axis)
is 1 if the axis is part of the path grouping.

This signal shows the status of the current block to be executed during block change. Subsequent status changes are ignored.

If the SERUPRO operation is ended with "Search target found",
DB31, ... DBX76.4 (path axis) refers to the target block.

9.8.2.1 Repositioning with controlled REPOS

Once SERUPRO has been used to find the target block, prior to continuing the interrupted program, a REPOS operation for repositioning the contour is performed. The REPOS mode "Reposition at the block start point of the target block" (RMBBL) is active by default. The REPOS mode can be defined user-specific via the NC/PLC interface:
DB21, ... DBX31.0 -.2 (REPOS mode A, B, C) (Page 1609)

REPOS mode RMNBL repositioning to the next point on the path

in the RMNBL REPOS mode, positioning is made for the REPOS start position from the next nearest contour point.

Example

The program is interrupted at any point in the block N110. The axes were then traversed to position (A), e.g. manually. Once SERUPRO has found target block N110, the REPOS operation with REPOS mode RMNBL is performed. With regard to the REPOS start position (A), point (B) is the next nearest point of the contour. The REPOS operation is completed when point (B) is reached. The programmed contour of the interrupted program is traversed again starting at point (B).
Specifying the REPOS mode via the NC/PLC interface

The REPOS mode can be specified via the following NC/PLC interface signal:
DB21, ... DBX31.0 - .2 (REPOS mode A, B, C)

Note

RMNBL is a general REPOS extension and it is not restricted to SERUPRO.

RMIBL and RMBBL behavior identically for SERUPRO.

DB21, ... DBX31.0 - .2 (REPOS mode) affects only the traversing motion of the path axes.

The behavior of the other axis can be changed individually using interface signal DB31, ... DBX10.0 (REPOSDELAY). The REPOS offset is not applied immediately, but only when it is next programmed.

For further information on the programming of the repositioning point, see:

References:
Programming Manual, Job Planning; Path Behavior, Section: Repositioning on contour

9.8.3 Accelerate block search

Machine data settings

The speed of execution of the SERUPRO operation can be influenced via the following machine data.

MD22600 $MC_SERUPRO_SPEED_MODE = <value>

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0       | Program test with block search/dry run feedrate:  
  - Axes: MD22601 $MC_SERUPRO_SPEED_FACTOR * dry run feedrate  
  - Spindles: MD22601 $MC_SERUPRO_SPEED_FACTOR * programmed speed  
  Dynamic limitations of axes / spindles are not considered. |
| 1       | Program test with programmed velocity:  
  - Axes: Dry run feedrate  
  - Spindles: Programmed speed  
  Dynamic limitations of axes / spindles are considered. |
### Program test with dry run feedrate

Under program test, traversing is performed with the programmed velocity / speed. Dynamic limitations of axes / spindles are considered.

### Program test with block search velocity

Under program test, traversing is performed with the following velocity:
- **Axes**: $\text{MD22601} \times \text{MC_SERUPRO_SPEED_FACTOR} \times \text{programmed feedrate}
- **Spindles**: $\text{MD22601} \times \text{MC_SERUPRO_SPEED_FACTOR} \times \text{programmed speed.}

Dynamic limitations of axes / spindles are **not** considered.

#### Note

When revolutional feedrate is active (e.g. G95), the programmed feedrate is not multiplied by $\text{MD22601} \times \text{MC_SERUPRO_SPEED_FACTOR}$ but only the programmed spindle speed. This results, here too, in an increase in the effective path velocity by $\text{MD22601} \times \text{MC_SERUPRO_SPEED_FACTOR}$.

### Supplementary conditions

#### Main axes

MD22600 $\text{MC_SERUPRO_SPEED_MODE}$ acts on the following main run axes with SERUPRO:
- PLC axes
- Command axes
- Positioning axes
- Reciprocating axes

#### Synchronized actions

**NOTICE**

**Actions of synchronized action may not be executed with SERUPRO**

Because during SERUPRO other actual values (e.g. axis positions) are generated internally that are different from those in normal program execution, with SERUPRO, conditions of synchronized actions that check the actual values (e.g. axis positions) are no longer recognized as true (TRUE) and that the action part of the synchronized action is therefore not executed.

### Revolutional feedrate

Effects of $\text{MD22601} \times \text{MC_SERUPRO_SPEED_FACTOR}$ during DryRun:
- **Switchover** of G95/G96/G961/G97/G971 to G94
- Tapping and thread cutting: Normal dry run velocity.
Rigid tapping

- With "rigid tapping" (G331/G332), the spindle is interpolated under position control in a path grouping. The drilling depth (linear axis), the thread pitch, and speed (spindle) are defined. During dry run, the velocity of the linear axis is specified, the speed remains constant, and the pitch is adjusted. After SERUPRO, a position for the spindle results that deviates from normal mode because the spindle has revolved fewer times in SERUPRO.

9.8.4 SERUPRO ASUB

SERUPRO ASUB special points

Special points should be noted for SERUPRO ASUB with regard to:

- Reference point approach: Referencing via part program G74
- Tool management: Tool change and magazine data
- Spindle ramp-up: On starting a SERUPRO ASUB

G74 reference point approach

If command G74 (reference point approach) is programmed between the program start and the search target, this will be ignored by the NC.

SERUPRO approach does not take this G74 command into account!

Tool management

If tool management is active, the following setting is recommended:

MD20310 $MC_TOOL_MANAGEMENT_MASK Bit 20 = 0

The tool management command generated during the SERUPRO operation is thus not output to the PLC!

The tool management command has the following effect:

- The NC acknowledges the commands automatically.
- No magazine data is changed.
- Tool data is not changed.

Exception:
The tool enabled during the test mode can assume 'active' state. In this way, the wrong tool may be on the spindle after the SERUPRO operation.

Remedy:
The user starts a SERUPRO ASUB that is actually traversed. Prior to the start, the user can start an ASUB that loads the correct tool.

SERUPRO operation: Functionality: In sequence steps 2. to 6.
SERUPRO ASUB: Functionality: The sequence of step 7.
In addition, machine data setting MD20310 $MC_TOOL_MANAGEMENT_MASK Bit 11 = 1 is required because the ASUB may have to repeat a T selection.

Systems with tool management and auxiliary spindle are not supported by SERUPRO!

**Example**

**Tool change subprogram**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC L6</td>
<td>Tool change routine</td>
</tr>
<tr>
<td>N500 DEF INT TNR_AKTUELL</td>
<td>Variable for active T number</td>
</tr>
<tr>
<td>N510 DEF INT TNR_VORWAHL</td>
<td>Variable for preselected T number</td>
</tr>
<tr>
<td>N520 STOPRE</td>
<td>; Determine current tool</td>
</tr>
<tr>
<td>N530 IF $P_ISTEST</td>
<td>; In program testing</td>
</tr>
<tr>
<td>N540 TNR_AKTUELL = $P_TOOLNO</td>
<td>; From the program context</td>
</tr>
<tr>
<td>N550 ELSE</td>
<td>; Otherwise, the tool of the spindle is read out.</td>
</tr>
<tr>
<td>N560 TNR_AKTUELL = $TC_MPP6[9998,1]</td>
<td>; Read tool T number on the spindle</td>
</tr>
<tr>
<td>N570 ENDF</td>
<td></td>
</tr>
<tr>
<td>N580 GETSELT(TNR_VORWAHL)</td>
<td>; Read T number of the preselected tool of the master spindle</td>
</tr>
<tr>
<td>N590 IF TNR_AKTUELL &lt;&gt; TNR_VORWAHL</td>
<td>; Execute tool change only if tool is not yet current</td>
</tr>
<tr>
<td>N600 G0 G40 G60 G90 SUPA X450 Y300 Z300 D0</td>
<td>; Approach tool change point</td>
</tr>
<tr>
<td>N610 M206</td>
<td>; Execute tool change</td>
</tr>
<tr>
<td>N620 ENDF</td>
<td></td>
</tr>
<tr>
<td>N630 M17</td>
<td></td>
</tr>
</tbody>
</table>

**ASUB for calling the tool change routine after type 5 block search**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC ASUPWZV2</td>
<td></td>
</tr>
<tr>
<td>N1000 DEF INT TNR_SPINDEL</td>
<td>; Variable for active T number</td>
</tr>
<tr>
<td>N1010 DEF INT TNR_VORWAHL</td>
<td>; Variable for preselected T number</td>
</tr>
<tr>
<td>N1020 DEF INT TNR_SUCHLAUF</td>
<td>; Variable for T number determined in block search</td>
</tr>
<tr>
<td>N1030 TNR_SPINDEL = $TC_MPP6[9998,1]</td>
<td>; Read tool T number on the spindle</td>
</tr>
<tr>
<td>N1040 TNR_SUCHLAUF = $P_TOOLNO</td>
<td>; Read T number determined by search, i.e. this tool determines the current tool offset</td>
</tr>
<tr>
<td>N1050 GETSELT(TNR_VORWAHL)</td>
<td>; Read T number of preselected tool</td>
</tr>
<tr>
<td>N1060 IF TNR_SPINDEL = TNR_SUCHLAUF GOTOF ASUP_ENDE1</td>
<td></td>
</tr>
<tr>
<td>N1070 T = $TC_TP2[TNR_SUCHLAUF]</td>
<td>; T selection by tool name</td>
</tr>
</tbody>
</table>
In both of the programs PROC L6 and PROC ASUPWZV2, the tool change is programmed with M206 instead of M6. ASUB "ASUPWZV2" uses different system variables to detect the progress of the program ($P_TOOLNO) and represent the current status of the machine ($TC_MPP6[9998,1] ).

**Spindle ramp-up**

When the SERUPRO ASUB is started, the spindle is not accelerated to the speed specified in the program because the SERUPRO ASUB is intended to move the new tool into the correct position at the workpiece after the tool change.

A spindle ramp-up is performed with SERUPRO ASUB as follows:

- SERUPRO operation has finished completely.
- The user starts the SERUPRO ASUB via function block FC 9 in order to ramp up the spindle.
- The start after M0 in the ASUB does not change the spindle status.
- SERUPRO ASUB automatically stops before the REPOS part program block.
- The user presses START.
- The spindle accelerates to the target block state if the spindle was not programmed differently in the ASUB.

**Note**

Modifications for REPOS of spindles:

The transitions of speed control mode and positioning mode must be taken into consideration in the event of modifications in SERUPRO approach and spindle functionality.

For further information on the operating mode switchover of spindles, see Section "Modes (Page 1200)".
9.8.5 Selfacting SERUPRO

Selfacting SERUPRO

The channel-specific function "Self-acting SERUPRO" allows a SERUPRO sequence without having to previously define a search target in a program of the associated SERUPRO channels.

In addition, a special channel, the "seruproMasterChan", can be defined for each "Self-acting SERUPRO". A search target can be defined in this channel.

The "Selfacting SERUPRO" function supports the SERUPRO cross-channel block search.

Function

The "SelfActing SERUPRO" operation cannot be used to find a search target. If the search target is not reached, no channel is stopped. In certain situations, however, the channel is nevertheless stopped temporarily. In this case, the channel will wait for another channel. Examples are: Wait marks, couplings, or axis replacement.

Wait phase occurs:
During this wait phase, the NC checks whether the channel "seruproMasterChan" has reached a search target. If no search target is reached, the Wait phase is left.

If the search target is reached, the SERUPRO operation is also ended in the channel. The "seruproMasterChan" channel must have been started in normal SERUPRO mode.

No wait phase occurs:
"Self-Acting SERUPRO" is ended by M30 of the part program. The channel is now in Reset state again. A SERUPRO approach does not not take place.

Starting a group of channels
If a group of channels is only started with "SelfActing SERUPRO", then all channels are ended with "RESET".

Exceptions:
A channel waits for a partner channel that has not been started at all.

A cross-channel block search can be carried out as follows:

- Via the HMI, the user selects the channels that must work together (channel group).
- The user chooses an especially important channel from the channel group for which he wants to select a search target explicitly (target channel).
- The HMI will then start SERUPRO on the target channel and "SelfActing SERUPRO" in the remaining channels of the channel group.

The operation is complete if each channel concerned has deleted "seruproActive". "Selfacting SERUPRO" accepts no master channel on another NCU.
**Activation**

"Self-acting SERUPRO" is activated via the HMI as a block search start for the Type 5 block search for target channel "seruproMasterChan".

No search target is specified for dependent channels started from the target channel.

**9.8.6 Locking a program section for "Continue machining at the contour"**

**Programmed interrupt pointer**

If because of manufacturing and/or process-related reasons, "Continue machining at the contour" may not be possible within a certain program section at a program abort, this program section can be locked for the target block of a block search.

If after a program abort there is a block search for the interruption point within the locked program section for "Continue machining at the contour", the last executable block (main run block) before the start of the locked section is used by the control as target block (hold block).

**Programming**

**Syntax**

`IPTRLOCK()`

**Functionality**

Marks the beginning of the program section as of which "Continue machining at the contour" is locked. The next executable block (main run block) in which `IPTRLOCK` becomes active is now used as target block for a block search with "Continue machining at the interruption point", until the release with `IPTRUNLOCK`. This block is referred to as the hold block in the following.

Effectiveness: Modal

**Syntax**

`IPTRUNLOCK()`

Marks the end of the program section locked for "Continue machining at the contour". As of the next executable block (main run block) in which `IPTRLOCK` becomes active, the current block is used again as target block for a block search with "Continue machining at the interruption point". This block is referred to as the release block in the following.

Effectiveness: Modal

**Example**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N010 IPTRLOCK()</td>
<td>; Locked area: Start</td>
</tr>
<tr>
<td>N020 R1=R1+1</td>
<td></td>
</tr>
<tr>
<td>N030 G4 F1</td>
<td>; Hold block</td>
</tr>
<tr>
<td>...</td>
<td>; Locked area</td>
</tr>
</tbody>
</table>
Boundary conditions

- **IPTRLOCK** acts within a program (*.MPF, *.SPF) at the most up to the end of the program (M30, M17, RET). **IPTRUNLOCK** implicitly becomes active at the end of the program.

- Multiple programming of **IPTRLOCK** within a program does not have a cumulative effect. With the first programming of **IPTRUNLOCK** within the program or when the end of the program is reached, all previous **IPTRLOCK** calls are terminated.

- If there is a subprogram call within a locked area, "Continue machining at the contour" is also locked for this and all following subprogram levels. The lock also cannot be cancelled within the called subprogram through explicit programming of **IPTRUNLOCK**.

Example: Nesting of locked program sections in two program levels

With the activation of the "Continue machining at the contour" lock in PROG_1, "Continue machining at the contour" is also locked for PROG_2 and all following program levels.

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC PROG_1</td>
<td>; Program 1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N010 IPTRLOCK()</td>
<td></td>
</tr>
<tr>
<td>N020 R1=R1+1</td>
<td></td>
</tr>
<tr>
<td>N030 G4 F1</td>
<td>; Hold block</td>
</tr>
<tr>
<td></td>
<td>; Locked area: Start</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N040 PROG_2</td>
<td>; Locked area</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N050 IPTRUNLOCK()</td>
<td></td>
</tr>
<tr>
<td>N060 R2=R2+2</td>
<td></td>
</tr>
<tr>
<td>N070 G4 F1</td>
<td>; Release block</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC PROG_2</td>
<td>; Program 2</td>
</tr>
<tr>
<td>N210 IPTRLOCK()</td>
<td>; Ineffective due to program 1</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N250 IPTRUNLOCK()</td>
<td>; Ineffective due to program 1</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N280 RET</td>
<td>; Ineffective due to program 1</td>
</tr>
</tbody>
</table>
Example 3: Multiple programming of IPTRLOCK

```plaintext
PROC PROG_1 ; Program 1...
N010 IPTRLOCK()
N020 R1=R1+1
N030 G4 F1 ; Hold block...
; Locked area: Start
N150 IPTRLOCK()
; Locked area...
N250 IPTRLOCK()
; Locked area...
N360 IPTRUNLOCK()
N370 R2=R2+2
N380 G4 F1 ; Release block...
```

System variable

The status of the current block can be determined via the system variable $P_IPTRLOCK:

<table>
<thead>
<tr>
<th>$P_IPTRLOCK</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>The current block is <strong>not</strong> within a program section locked for “Continue machining at the contour”</td>
</tr>
<tr>
<td>TRUE</td>
<td>The current block is within a program section locked for “Continue machining at the contour”</td>
</tr>
</tbody>
</table>

Automatic function-specific "Continue machining at the contour" lock

For various couplings, the activation/deactivation of the “Continue machining at the contour” lock can be performed automatically channel-specifically with the activation/deactivation of the coupling:

MD22680 $MC_AUTO_IPTR_LOCK, bit x

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Electronic gear (EGON / EGOF)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Automatic &quot;Continue machining at the contour&quot; lock is active</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Automatic &quot;Continue machining at the contour&quot; lock is not active</td>
</tr>
<tr>
<td>1</td>
<td>Axial master value coupling (LEADON / LEADOF)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Automatic &quot;Continue machining at the contour&quot; lock is active</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Automatic &quot;Continue machining at the contour&quot; lock is not active</td>
</tr>
</tbody>
</table>

This program section begins with the last executable block before the activation and ends with the deactivation.
The automatic interrupt pointer is not active for couplings that were activated or deactivated via synchronized actions.

**Example:** Automatically declaring axial master value coupling as search-suppressed:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 G0 X100</td>
<td></td>
</tr>
<tr>
<td>N200 EGON(Y,&quot;NOC&quot;,X,1,1)</td>
<td>; Search-suppressed program section starts.</td>
</tr>
<tr>
<td>N300 LEADON(A,B,1)</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N400 EGOF(Y)</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N500 LEADOF(A,B)</td>
<td>; Search-suppressed program section ends.</td>
</tr>
<tr>
<td>N600 G0 X200</td>
<td></td>
</tr>
</tbody>
</table>

A program abort within search-suppressed program section (N200 - N500) always provides the interrupt pointer with N100.

**NOTICE**

**Unwanted state caused by function overlappings**

If there is an overlap of the "Programmable interrupt pointer" and "Automatic interrupt pointer" functions via machine data, the NC selects the largest possible search-suppressed area.

A program may need a coupling for almost all of the runtime. In this case, the automatic interrupt pointer would always point to the start of the program and the SERUPRO function would in fact be useless.

**9.8.7 Behavior during POWER ON, mode change and RESET**

SERUPRO is inactive during POWER ON. The mode change is permitted during SERUPRO. RESET will cancel SERUPRO and deselects the internally selected program test. SERUPRO cannot be combined with other block search types.
9.8.8 Supplementary conditions

9.8.8.1 STOPRE in the target block

STOPRE block

The STOPRE block receives all modal settings from the preceding block and can, therefore, apply conditions in advance in relation to the following actions:

- Synchronize program line currently processing with the main run.
- Derive modal settings for SERUPRO in order, for example, to influence this REPOS motion on approach of SERUPRO.

Example 1:

Position a Z axis by specifying an X axis setpoint.

When block "G1 F100 Z=$AA_IM[X]" is interpreted, the preceding STOPRE block ensures synchronization with the main run. The correct setpoint of the X axis is thus read via $AA_IM to move the Z axis to the same position.

Example 2:

Read and correctly calculate external zero offset.

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G1 X1000 F100 ;</td>
<td></td>
</tr>
<tr>
<td>N20 G1 X1000 F500 ;</td>
<td></td>
</tr>
<tr>
<td>N30 G1 X1000 F1000 ;</td>
<td></td>
</tr>
<tr>
<td>N40 G1 X1000 F5000 ;</td>
<td></td>
</tr>
<tr>
<td>N50 SUPA G1 F100 X200 ; move external zero offset after 200</td>
<td></td>
</tr>
<tr>
<td>N60 G0 X1000 ;</td>
<td></td>
</tr>
<tr>
<td>N70 ... ;</td>
<td></td>
</tr>
</tbody>
</table>

Via an implicit STOPRE before N50, the NCK can read and correctly calculate the current zero offset.

For a SERUPRO operation on the N50 search target, repositioning is on the implicit STOPRE in the SERUPRO approach and the velocity is determined from N40 with F5000.
Implicit preprocessing stop

Situations in which Interpreter issues an implicit preprocessing stop:

1. In all blocks in which one of the following variable access operations occurs:
   - Programming of a system variable beginning with $A...
   - Redefined variable with attribute SYNR/SYNRW

2. For the following part program commands:
   - Part program command MEACALC, MEASURE
   - Programming of SUPA (suppress frames and online offsets)
   - Programming of CTABDEF (start of curve table definition)
   - Part program command WRITE/DELETE (write/delete file)
   - Before the first WRITE/DELETE command in a sequence of such commands
   - Part program command EXTCALL
   - Part program command GETSELT, GETEXET
   - Tool change and active fine tool offset FTOCON

3. For the following command execution:
   - Finishing of Type 1 search ("Search without calculation") and
     Type 2 search with calculation ("Search at contour end point")

   Note: Type 2 search "Block search at contour start point" has the same behavior.

9.8.8.2 SPOS in target block

SPOS

If a spindle is programmed with M3/M4 and the target block contains an SPOS command, the spindle is switched over to SPOS on completion of the SERUPRO process (search target located). This is indicated on the VDI interface.

9.8.8.3 Travel to fixed stop (FXS)

During repositioning (REPOS), the "Travel to fixed stop" function (FXS) is repeated automatically. Every axis is taken into account. The torque programmed last before the search target is used as torque.

System variable

The system variables for "Travel to fixed stop" have the following meaning with SERUPRO:

- $AA_FXS: Progress of the program simulation
- $VA_FXS: Real machine state

The system variables always have the same values outside of the SERUPRO function.

ASUB

A user-specific ASUB can be activated for SERUPRO.
9.8.8.4 Travel with limited torque/force (FOC)

During repositioning (REPOS), the "Travel with limited torque/force" function (FOC) is repeated automatically. Every axis is taken into account. The torque programmed last before the search target is used as torque.

System variable

The system variables for "Travel with limited torque/force" have the following meaning with SERUPRO:

- $AA_FOC: Progress of the program simulation
- $VA_FOC: Real machine state

Supplementary condition

A changing torque characteristic cannot be implemented during repositioning.

Example

A program traverses axis X from 0 to 100 and switches "Travel with limited torque/force" (FOC) on every 20 increments for 10 increments. This torque characteristic is usually generated with non-modal FOC and cannot be performed during repositioning (REPOS). Instead, axis X is traversed from 0 to 100 with or without limited torque/force in accordance with the last programming.

9.8.8.5 Synchronous spindle

The synchronous spindle can be simulated.

The synchronous spindle operation with main spindle and any number of following spindles can be simulated in all existing channels with SERUPRO.

For further information about synchronous spindles, see:

References:

Function Manual, Extended Functions; Synchronous Spindle (S3)
9.8.8.6 Couplings and master-slave

Setpoint and actual value couplings

The SERUPRO operation is a program simulation in Program Test mode with which setpoint and actual value couplings can be simulated.

Specifications for EG simulation

For simulation of EG, the following definitions apply:
1. Simulation always takes place with setpoint coupling.
2. If not all leading axes are under SERUPRO, the simulation is aborted with Alarm 16952 "Reset Clear/No Start". This can occur with cross-channel couplings.
3. Axes that have only one encoder from the NCK point of view and are moved externally, cannot be simulated correctly. These axes must not be integrated in couplings.

**CAUTION**

Incorrect simulation

In order to be able to simulate couplings correctly, they must have been switched off previously.

This can be performed with machine data MD10708 $MA_SERUPRO_MASK.

Specifications for coupled axes

The SERUPRO operation simulates coupled axes always assuming that they are setpoint couplings. In this way, the end points are calculated for all axes that are used as target points for SERUPRO approach. The coupling is also active with "Search target found". The path from the current point to the end point is carried out for SERUPRO approach with the active coupling.

**LEADON**

The following specifications apply for the simulation of axial master value couplings:

1. Simulation always takes place with setpoint coupling.
2. SERUPRO approach takes place with active coupling and an overlaid motion of the following axis in order to reach the simulated target point.

The following axis that is moved solely by the coupling cannot always reach the target point. In SERUPRO approach, an overlaid linear motion is calculated for the following axis to approach the simulated point!
Reaching simulated target point for LEAD with JOG

At the time of "Search target found", the coupling is already active, especially for the JOG motions. If the target point is not reached, SERUPRO approach can be used to traverse the following axis with active coupling and an overlaid motion to the target point.

Note

For further information on the repositioning of axis couplings, see Section "Repositioning to the contour (REPOS) (Page 497)".

Master-slave

A system ASUB can be started automatically after the block search is finished. In this subprogram, the user can control the coupling state and the associated axis positions subsequently. The required information is provided via the following system variables:

<table>
<thead>
<tr>
<th>System variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_SEARCH_MASLD [&lt;slave axis&gt;]</td>
<td>Position offset between slave and master axis when the link is closed.</td>
</tr>
<tr>
<td>$AA_MASL_STAT [&lt;slave axis&gt;]</td>
<td>Current state of a master/slave coupling</td>
</tr>
<tr>
<td>$P_SEARCH_MASLC [&lt;slave axis&gt;]</td>
<td>Status: The state of the coupling was changed during the block search</td>
</tr>
</tbody>
</table>

The system variables are deleted when the coupling is switched on with MASLON.

Note

The coupled axes must be in the same channel when the block search is executed.

References

Further information on the master-slave coupling can be found in:

Function Manual, Special Functions; Section "TE3: Speed/torque coupling, master-slave"
Example

- **System ASUB**
  - Path and name: `/N_CMA_DIR/PROGEVENT.SPF`
  - Master axis: X
  - Slave axis: Y

**Program code**

| N10 IF(($S_SEARCH_MASLC[Y]< >0) AND ($AA_MASL_STAT[Y]< >0)) |
| N20 MASLOF(Y) |
| N30 SUPA Y = $AA_IM[X] - $P_SEARCH_MASLD[Y] |
| N40 MASLON(Y) |
| N50 ENDIF |
| N60 REPOSA |
| ... |
| RET |

- **Machine data**
  To ensure that the ASUB starts automatically, the following machine data must be set:
    - NC-specific machine data:
      - MD11604 $MN_ASUP_START_PRIO_LEVEL = 100
      - MD11450 $MN_SEARCH_RUN_MODE = 'H02'
    - Channel-specifically for the channel in which the ASUB is started or generally for all channels:
      - MD20105 $MC_PROG_EVENT_IGN_REFP_LOCK, bit<n> = TRUE
        n: For all required event-driven program calls (prog events)
      - MD20115 $MC_IGNORE_REFP_LOCK_ASUP, bit<n> = TRUE
        n: For all **required user** interrupts

**NOTICE**

**System interrupts**

Interrupts 9 and 10 used by the system are released with MD20115
$MC_IGNORE_REFP_LOCK_ASUP, bit 8 and bit 9. Interrupt 9 starts an ASUB which contains traversing motions.
**Axis couplings**

- Acceleration of the processing speed and leading axis and following axes in different channels
  For a leading axis whose following axes are assigned to a different channel than that of the leading axis, the setting for acceleration of the processing speed has no effect (MD22601 $MC_SERUPRO_SPEED_FACTOR):

- Coupled motion
  The coupled motion function (TRAILON) is supported by SERUPRO.
  For further information on coupled motion with TRAILON and TRAILOF, see:
  
  References:
  - Function Manual, Special Functions; Axis Couplings (M3)
  - Programming Manual, Job Planning; Section "Axis couplings"

- Gantry axes
  The gantry axis function is supported by SERUPRO.
  For further information on the functionality of gantry axes, see:
  
  References:
  - Function Manual, Special Functions; Section "G1: Gantry axes"

- Tangential control
  The tangential follow-up of individual axes function is supported by SERUPRO.
  Additional information about the tangential control can be found in:
  
  References:
  - Function Manual, Special Functions; Section "T3: Tangential control"

**9.8.8.7 Axis functions**

**SERUPRO conditions**

The special conditions for SERUPRO must be observed with axis enable, autonomous axis operations, and axis replacement.

**Axis enable**

The axial interface DB31, ... DBX3.7 ("Program test axis/spindle enable") controls the axis enables if no closed-loop controller enable is to (or can) be issued at the machine and is active only during the program test or when SERUPRO is active.

It is possible to issue this enable via interface signal PLC→NCK DB31, ... DBX3.7 (program test axis/spindle enable). If the real servo enable is missing during program test or SERUPRO, the effect on the axes/spindles is as follows:

- As soon as the simulated program run intends to move an axis/spindle, the message "Waiting for axis enable" or "Waiting for spindle enable" is displayed and the simulation is stopped.

- If during a simulated motion, NC/PLC interface signal DB31, ... DBX3.7 (program test axis/spindle enable) is then canceled, alarm 21612: "Channel %1 axis %2 NC/PLC interface signal 'controller enable' reset during motion" is activated.
Autonomous axis operations

Autonomous single-axis operations are axes controlled by the PLC that can also be simulated on SERUPRO. During SERUPRO operation, as in normal operation, the PLC can take over or give up control of an axis. If required, this axis can also be traversed using FC18. The PLC takes over control of the axis before the approach block and is responsible for positioning this axis. This is valid for all block search types.

For further information about autonomous single-axis operations, see:

References:
Function Manual, Extended Functions; Positioning Axes (P2)

Axis replacement

Problem: A program moves an axis and gives up control before the target block with WAITP(X). X is thus not subject to REPOS and the axis is not taken into account in SERUPRO approach.

Via the machine data MD11470 $MN_REPOS_MODE_MASK, the following behavior can be achieved for SERUPRO-REPOS:

The neutral axes are moved as "command axes" in the SERUPRO-REPOS. The axis interpolates without a path context even if it was last programmed as a path axis. In this scenario, the velocity results from MD32060 $MA_POS_AX VELO. After SERUPRO approach, this axis is again neutral.

Neutral axes that are however not allowed to be repositioned must receive the axial NC/PLC interface signal "REPOSDELAY". This deletes the REPOS movement.

Example:

After SERUPRO, one axis is deliberately moved in the synchronized action via technology cycles. The command axes are always moved in the approach block, never in the target block. The target block can only be changed if all command axes have been moved to the end.

⚠️ CAUTION

The PLC-controlled axis is not repositioned

Aaxes enabled by RELEASE(X) before the target block are not repositioned.

9.8.8.8 Gear stage change

Operational sequences

The gear stage change (GSC) requires physical motions from the NCK in order to be able to engage a new gear.

In the SERUPRO operation, no gear stage change is required and is carried out as follows:

Some gears can only be changed when controlled by the NC, since either the axis must oscillate or a certain position must be approached beforehand.

The gear stage change can be suppressed selectively for DryRun, program test, and SERUPRO using bits 0 to 2 in MD35035 $MA_SPIND_FUNCTION_MASK.
The gear stage change must then be performed in REPOS; this will work even if the axis involved is to be in "speed control mode" at the target block. In other cases, the automatic gear stage change is denied with an alarm if, for example, the axis was involved in a transformation or coupling between the gear stage change and the target block.

**Note**
For further information on gear stage changes in DryRun, Program test and SERUPRO, see Section "S1: Spindles (Page 1199)".

### 9.8.8.9 Superimposed motion

**Only SERUPRO**

If "overlaid movements" are used, only the block search via program test (SERUPRO) can be used, since the overlaid movements are interpolated accordingly in the main run. This applies in particular to $AA_OFF.

**Velocity profile instead of maximum axis velocity**

During Program test, a velocity profile must be used, which allows "superimposed movements" to be interpolated during the main run. It is thus not possible to interpolate at the maximum axis velocity.

The axis velocity is set in "Dry run feedrate" mode using SD42100 $SC_DRY_RUN_FEED.

The velocity of the SERUPRO operation is selected using MD22600 $MC_SERUPRO_SPEED_MODE.

### 9.8.8.10 NC/PLC interface signals

**REPOS offset available**

If a REPOS offset has resulted for an axis during SERUPRO, this is displayed via the axial NC/PLC interface at the end of the SERUPRO operation:

DB31, ...DBX70.0 == 1 (REPOS offset available)

**Validity of the REPOS offset**

The REPOS offset becomes invalid at the start of a SERUPRO ASUB or NC start to resume the machining:

DB31, ...DBX70.1 == 1 (REPOS offset invalid)

The axis can be traversed manually in JOG mode or via the PLC user program using FC 18 between the end of the SERUPRO operation and NC start to resume the machining. If the REPOS offset is traversed completely, the interface signal is reset.
9.8.8.11 Making the initial settings more flexible

Initial setting/initial SERUPRO setting

Machine data MD20112 $MC\_START\_MODE\_MASK defines the initial setting of the control for part program start with respect to the G codes (especially the current plane and settable zero offset), tool length compensation, transformation, and axis couplings. The special option exists for the SERUPRO operation of using MD22620 $MC\_ENABLE\_START\_MODE\_MASK\_PRT to select an initial setting that differs from the normal part program start. The new setting must therefore been stored in:

MD22620 $MC\_START\_MODE\_MASK\_PRT

The meaning of the bits of MD22620 is identical to those of:

MD20112 $MC\_START\_MODE\_MASK.

Example:

The synchronous spindle coupling at the beginning of the SERUPRO operation is retained for the part program start.

```plaintext
$MC\_START\_MODE\_MASK = 'H400'
$MC\_START\_MODE\_MASK\_PRT = 'H00'
$MC\_ENABLE\_START\_MODE\_MASK\_PRT = 'H01'

; synchronous spindle coupling not configured
; is switched off
; remains active

$MC\_START\_MODE\_MASK\_PRT is evaluated in SERUPRO instead of $MC\_START\_MODE\_MASK
```

9.8.8.12 Compressor functions (COMPON, COMPCURV, COMPCAD)

- If the target block for block search type 2 or type 4 (block search with calculation to ...) is in a program section in which a compressor function (COMPON, COMPCURV, COMPCAD) is active, positions are approached on the path calculated by the compressor on repositioning. These positions must precisely match the positions on the path programmed in the part program.

- If programmed blocks are eliminated from the part program during compression, these blocks will not be found at the target block in the block search ⇒ Alarm 15370 "Search target not found."
9.8.9 System variable

Overview of the system variables relevant for SERUPRO:

<table>
<thead>
<tr>
<th>System variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_ASUP, bit 20</td>
<td>ASUB activation reason: $AC_ASUP, Bit 20 == 1 ⇒ system ASUB active, reason: SERUPRO search goal reached</td>
</tr>
<tr>
<td>$AC_SERUPRO</td>
<td>SERUPRO status: $AC_SERUPRO == 1 ⇒ SERUPRO is active</td>
</tr>
<tr>
<td>$P_ISTEST</td>
<td>Program test status: SERUPRO active ⇒ $P_ISTEST == 1</td>
</tr>
<tr>
<td>$P_SEARCHL</td>
<td>Most recently active block-search type: $P_SEARCHL == 5 from the start of SERUPRO to reset or end of program</td>
</tr>
<tr>
<td>$AC_REPOS_PATH_MODE</td>
<td>REPOS mode for repositioning the contour after a SERUPRO</td>
</tr>
</tbody>
</table>

9.9 Program operation

PLC, MD, operation

The execution of part programs can be controlled via the HMI in many ways using PLC inputs, machine data settings and operator inputs.

Definition

The execution of part programs or part program blocks in AUTOMATIC or MDA modes is referred to as program operation.

Channel control

Every channel can be manipulated by means of interface signals from the PLC. The control is exercised via mode groupspecific or channelspecific interface signals. An overview of these signals is given under data lists in this Description of Functions.

Status messages

Each channel reports its current program operation status to the PLC with interface signals. These signals are, in turn, divided up into mode groupspecific and channelspecific signals.
9.9.1 Initial settings

Machine data

Defined conditions can be set via machine data for the program operation or certain implementations of the NC language scope.

Initial settings

Initial settings can be programmed in channel-specific machine data for each channel. These initial settings affect, for example, G groups and auxiliary function output.

Auxiliary function output

The timing for output of auxiliary functions can be predefined via machine data AUXFU_x_SYNC_TYPE (MD22200, 22210, 22220, 22230, 22240, 22250, 22260), (output timing for M, S, T, H, F, D, E functions). For more detailed explanations, see Section "H2: Auxiliary function outputs to PLC (Page 377)".

G groups

An initial programming setting can be specified for each of the available G groups using MD20150 $MC_GCODE_RESET_VALUES (reset state of G groups). This initial setting is automatically active during program start or in Reset until it is deselected by a G command from the same G group.

Via the MD22510 $MC_GCODE_GROUPS_TO_PLC (G codes, which are output to interface NCK-PLC after block change / RESET), the output of the G codes to the PLC interface can be activated.

A list of G groups with the associated G functions is available in:

References:
Programming Manual, Fundamentals

Basic configurations of the NC language scope for SINUMERIK solution line

For SINUMERIK 840D sl, certain basic configurations of the NC language scope can be generated (configurable) via machine data. The options and functions of the NC language scope is specially tailored (configured) to the needs of the user.

NC language scope

The way that non-active options and functions should be moved with NC language commands can be set via the machine data MD10711 $MN_NC_LANGUAGE_CONFIGURATION:

0: All available language commands can be programmed. Whether or not the needed function is activated can only be recognized upon execution.

If only certain options are enabled and not all operations are available:

1: All the language commands are known. Language commands for non-enabled options are already recognized at the beginning of the program interpretation and lead to the alarm 12553 "option/function is not active."

K1: Mode group, channel, program operation, reset response
9.9 Program operation
2: Only those language commands are known which correspond to the current scope of enabled options of the NCK software. All commands for non-enabled options are not recognized and trigger the alarm 12550 "Name not defined or option/function not available".

Note
Option-free functions also have the status "enable option"

If only certain functions are activated:

3: All the language commands are known. Non-activated functions are already recognized at the beginning of the program interpretation and result in the alarm 12553 "Option/function is not active". For example, if the option date is set for the cylinder surface transformation, but the transformation is not activated in machine data MD24100 $MC_TRAOF_TYPE_1, then the programming of TRACYL triggers the alarm 12553.

4: Only those NC language commands are known which correspond to the current scope of active functions of the NCK software. All commands for non-active functions are not recognized and trigger the alarm 12550 "Name not defined or option/function not available". Whether the command in question is generally unavailable in the Siemens NC language or whether this is true only on the corresponding system cannot be distinguished in this scenario.

Whether the current NC language scope of enabled options and active functions is also truly programmable can be checked using the STRINGIS program command, see example.

Check sample application for NC language scope on cylinder surface transformation TRACYL

The cylinder surface transformation is optional and must be enabled beforehand. In order to check this, the following initial conditions are assumed:

The cylinder surface transformation option is not enabled and the machine data
$MN_NC_LANGUAGE_CONFIGURATION = 2; NC language command TRACYL is unknown

The following program is started

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1 R1=STRINGIS(&quot;TRACYL&quot;)</td>
<td>; R1 is 0 (TRACYL is an unknown name)</td>
</tr>
<tr>
<td>N2 IF STRINGIS(&quot;TRACYL&quot;)==204</td>
<td></td>
</tr>
<tr>
<td>N3 TRACYL(1, 2, 3)</td>
<td>; Block is not interpreted</td>
</tr>
<tr>
<td>N4 ELSE</td>
<td></td>
</tr>
<tr>
<td>N5 G00</td>
<td></td>
</tr>
<tr>
<td>N6 ENDIF</td>
<td></td>
</tr>
<tr>
<td>N7 M30</td>
<td></td>
</tr>
</tbody>
</table>

Example of whether STRINGIS result is programmable or not

The result of STRINGIS = number-coded return value (three-digit)
Number coding of the basic information (1st digit from the left):
000 Name is unknown, programming is denied with Alarm 12550
100: Name is known, but cannot be programmed, triggers alarm 12533
200: Name/symbol is known, but interpretation is not possible

2xx: Name/symbol is known, the command can be programmed, if xx > 0

Definition for name/symbol:

Name: Any STRING that is checked to see whether it is a component of the NC language in the existing NCK version or configuration.

Symbol: Contains the description or significance of an NC language command that is needed for the NC program and cycle interpretation.

Dependent on machine data MD10711

$MN_NC_LANGUAGE_CONFIGURATION = (set value) results in the following interpretations of the option and function relative to their programmability 2xx:

<table>
<thead>
<tr>
<th>MD10711</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
<td>Function</td>
<td>Return value as the basic information (1st digit from the left)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Definition for option/function:

0 corresponds to option not activated or function deactivated
1 corresponds to option activated or function activated

For more detailed information on the value ranges of 2xx programmable functions, see:

References:
Programming Manual, Job Planning, Additional Functions,
Section: Check scope of NC language present (STRINGIS)

9.9.2 Selection and start of part program or part program block

Reset status

Channel status
A part program can be selected only if the relevant channel is in the Reset state.

Start command, channel status
There are two possible START commands for initiating processing of a part program or part program block:

- The channel-specific interface DB21, ... DBX7.1 (NC Start), which is usually controlled from the machine control panel key NC Start, starts program execution in the same channel.
- With the NC instruction START, program execution in the first channel can be started from the second channel, for example.
The START command can only be executed in AUTOMATIC and MDA modes. For this, the channel concerned must be in the following state:

DB21, ... DBX35.7 (channel status reset) or
DB21, ... DBX35.6 (channel status interrupted).

**Signals, Alarms**

**Required signal states**

The part program can now be enabled for execution in the channel with the START command on the condition that certain signal states exist on the machine.

The following enable signals are relevant on the NC/PLC interface:

- DB11 DBX4.4 (mode group ready) must be present
- DB11 DBX0.7 (mode group reset) must not be present
- DB21, ... DBX1.7 (activate program test) must not be present
- DB21, ... DBX7.0 (NC start disable) must not be present
- DB21, ... DBX7.2 (NC stop at the block limit) must not be present
- DB21, ... DBX7.3 (NC stop) must not be present
- DB21, ... DBX7.4 (NC stop axes plus spindle) must not be present
- DB21, ... DBX7.7 (reset) must not be present
- DB10 DBX56.1 (emergency stop) may not be present
- No axis or NCK alarm must be active

For a further explanation of the individual signals see Section 5.

**Execution of command**

The part program or the part program block is automatically executed and the following interface signals are set:

DB21, ... DBX35.5 (channel status reset)
DB21, ... DBX35.0 (program status running)

The program is processed until the end of the program has been reached or the channel is interrupted or aborted by a STOP or RESET command.

**Alarms**

Under certain conditions the START command will have no effect and one of the following alarms will be triggered:

- 10200 "No NC Start permitted with active alarm"
- 10202 "No NC Start permitted with active command"
- 10203 "No NC Start permitted for non-referenced axes"

**References:**

Diagnostics Manual, Alarms
9.9.3 Part program interruption

Channel status

A part program interruption is only executed when the channel and program are active:

- DB21, ... D35.5 == 1 ("Channel active")
- DB21, ... D35.0 == 1 ("Program running")

STOP commands

The part program processing can be interrupted via the following STOP commands:

- DB21, ... DBX7.2 ("NC stop at the block limit")
- DB21, ... DBX7.3 ("NC stop")
- DB21, ... DBX7.4 ("NC stop, axes plus spindles")
- DB21, ... DBX2.0 ("Single block")
- Program command M00 or M01

The channel and program are then in the state "interrupted":

- DB21, ... D35.6 == 1 ("Channel interrupted")
- DB21, ... D35.3 == 1 ("Program interrupted")

References

A detailed description of the interface signals can be found in:
Function Manual Basic Functions; NC/PLC Interface Signals (Z1)

Sequence

The following actions are performed after a STOP command:

- Interruption of the part program execution at the next block limit (with "NC stop at the block limit", M00, M01 or single block), or immediately for all other STOP commands.
- The traversing axes of the channel are stopped via braking ramp. The braking of the axes down to standstill can be extended over several blocks.
- The block indicator shows the current block at the point of interruption.
- The auxiliary functions that have not been output before the point of interruption are no longer output.
Possible actions in the interrupt state
Various functions can be performed in the channel during a part program interruption, for example:

- Overstore
  References: Operating Manual, HMI Advanced, Section "Machine operating area" > Automatic mode" > "Overstore"

- Block search
  References: Function Manual, Basic Functions; Section "Mode group, channel, program operation, reset response (K1)" > "Block search" or "Block search type 5 SERUPRO"

- Repositioning (REPOS)
  References: Function Manual, Basic Functions; Section "Mode group, channel, program operation, reset response (K1)" > "Block search type 5 SERUPRO" > "REPOS" > "Repositioning with controlled REPOS"

- Oriented tool retraction
  References:
  - Programming Manual, Job Planning; Section "Tool offsets"
  - Description of Functions, Basic Functions; Section "Tool offsets (W1)" > "Orientable toolholders" > ""

- Interrupt routine (see Section "Asynchronous subprograms (ASUBs), interrupt routines (Page 558)"

- DRF function, offset of the workpiece zero
  References: Function Manual, Extended Functions; Manual and Handwheel Travel (H1)

- Starting the interrupted program via:
  - START command from another channel
    References: Programming Manual, Job Planning; Section "Flexible NC programming" > "Program coordination (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM)"
  - NC/PLC interface:
    DB21, ... DBX7.1 (NC start)

9.9.4 RESET command

Command priority

Channel status
The RESET command can be executed in every channel state. This command is aborted by another command.
## Commands

### RESET-Command

The following Reset commands are available:

- DB11, ... DBX0.7 ("mode group reset")
- DB21, ... DBX7.7 ("Reset")

For a further explanation of the individual interface signals, please see References: /FB1/ Function Manual Basic Functions; NC/PLC interface signals (Z1)

A RESET command can be used to interrupt an active part program or a part program block (in MDA).

After execution of the Reset command, the interface signal DB21, ... DBX35.7 ("Channel status Reset") is set.

The part program cannot be continued at the point of interruption. All the axes of the channel go into exact stop unless they are in followup mode. The same applies to the spindles configured in the channel.

The following actions are executed when the RESET command is triggered:

- Part program preparation is stopped immediately.
- Axes and, if they exist, spindles in the channel are decelerated along a braking ramp.
- Any auxiliary functions of the current block not yet output, are no longer output.
- The block indicator is reset to the beginning of the part program.
- All Reset alarms (channelspecific, axispecific, spindlespecific) are cleared from the display.

### 9.9.5 Program status

The status of the selected program is displayed in the interface for each channel. The PLC can then trigger certain responses and interlocks configured by the manufacturer depending on the status at the interface. The program status is only displayed in AUTOMATIC and MDA mode. In all other modes the program status is aborted or interrupted.

#### Program states

The following program states are displayed at the NC/PLC interface (DB21, ... ):

- DB21, ... DBX35.4 ("aborted")
- DB21, ... DBX35.3 ("interrupted")
- DB21, ... DBX35.2 ("stopped")
- DB21, ... DBX35.1 ("waiting")
- DB21, ... DBX35.0 ("running")

A detailed description of the interface signals can be found in Section "Signals from channel (DB21, ... ) (Page 1613)."
Effects of commands and NC/PLC interface signals

The program state of an active program is influenced by various commands and NC/PLC interface signals. The following table shows the respective resulting state.

<table>
<thead>
<tr>
<th>Command or NC/PLC interface signal (IS)</th>
<th>Resulting state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aborted</td>
<td></td>
</tr>
<tr>
<td>Interrupted</td>
<td></td>
</tr>
<tr>
<td>Stopped</td>
<td></td>
</tr>
<tr>
<td>Waiting</td>
<td></td>
</tr>
<tr>
<td>Running</td>
<td></td>
</tr>
</tbody>
</table>

- **IS "Reset"**
- **IS "NC stop"**
- **IS "NC stop at block limit"**
- **IS "NC stop axes and spindles"**
- **IS "Read-in disable"**
- **IS "Feed stop, channelsp."**
- **IS "Feed stop, axissp."**
- **Feed override = 0%**
- **IS "Spindle stop"**
- **M02/M30 in a block**
- **M00/M01 in a block**
- **IS "Single block"**
- **IS "Delete distance-to-go"**
- **Auxiliary functions output to PLC but not yet acknowledged**
- **Wait instruction in program**

9.9.6 Channel status

The current channel status is displayed in all operating modes at the NC/PLC interface (DB21, ...) for each channel.

Channel states

The following channel states are displayed at the NC/PLC interface (DB21, ...):

- DB21, ... DBX35.7 ("reset")
- DB21, ... DBX35.6 ("interrupted")
- DB21, ... DBX35.5 ("active")

A detailed description of the interface signals can be found in Section "Signals from channel (DB21, ...)
(Page 1613)."
**Effects of commands and NC/PLC interface signals**

The channel state of an active program is influenced by various commands and NC/PLC interface signals. The following table shows the respective resulting state.

<table>
<thead>
<tr>
<th>Command or NC/PLC interface signal (IS)</th>
<th>Resulting state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>x</td>
</tr>
<tr>
<td>IS &quot;NC stop&quot;</td>
<td>x</td>
</tr>
<tr>
<td>IS &quot;NC stop at block limit&quot;</td>
<td>x</td>
</tr>
<tr>
<td>IS &quot;NC stop axes and spindles&quot;</td>
<td>x</td>
</tr>
<tr>
<td>IS &quot;Read-in disable&quot;</td>
<td>x</td>
</tr>
<tr>
<td>IS &quot;Feed stop, channelsp.&quot;</td>
<td>x</td>
</tr>
<tr>
<td>IS &quot;Feed stop, axissp.&quot;</td>
<td>x</td>
</tr>
<tr>
<td>Feed override = 0%</td>
<td></td>
</tr>
<tr>
<td>IS &quot;Spindle stop&quot;</td>
<td>x</td>
</tr>
<tr>
<td>M02/M30 in a block</td>
<td>x</td>
</tr>
<tr>
<td>M00/M01 in a block</td>
<td>x</td>
</tr>
<tr>
<td>IS &quot;Single block&quot;</td>
<td>x</td>
</tr>
<tr>
<td>IS &quot;Delete distance-to-go&quot;</td>
<td>x</td>
</tr>
</tbody>
</table>

The "Channel state active" signal is obtained when a part program or part program block is being executed or when axes are traversed in JOG mode.

### 9.9.7 Responses to operator or program actions

**Status transitions**

The following table shows the channel and program states that result after certain operator and program actions.

The left-hand side of the table shows the channel and program states and the mode groups from which the initial situation can be selected. Various operator/program actions are listed on
the righthand side of the table, the number of the situation after the action has been carried out is shown in brackets after each action.

Table 9-3  Responses to operator or program actions

<table>
<thead>
<tr>
<th>Situation</th>
<th>Channel status</th>
<th>Program status</th>
<th>Active mode</th>
<th>Operator or program action (situation after the action)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>U</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Channel status: R --> aborted, U --> interrupted, A --> running
Program status: N --> aborted, U --> interrupted, S --> stopped, W --> waiting
Modes: A --> aborted, M --> aborted, J --> aborted, A --> running
9.9.8 Part-Program Start

Start handling

Table 9-4 Typical program sequence

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Command</th>
<th>Conditions (must be satisfied before the command)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Load program (via the operator interface or part program)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Select AUTOMATIC mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Program preselection</td>
<td>Channel preselected \ Preselected channel in RESET state \ User ID sufficient for program preselection</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>NC start for preselected channel</td>
<td>NC start disable not available \ Reference point approached in all axes</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>Program execution</td>
</tr>
<tr>
<td>6</td>
<td>M02/M30/RESET</td>
<td>None</td>
<td>End of program</td>
</tr>
</tbody>
</table>
9.9.9 Example of a timing diagram for a program run

![Timing Diagram]

Figure 9-6 Examples of signals during the program run

9.9.10 Program jumps

9.9.10.1 Jump back to start of program

Function

With the function "Jump back to start of the program" the control jumps back from a part program to the beginning of the program. The program is then processed again.

As compared to the function "Program jumps to jump marks", with which a repeated processing of the program can also be implemented, the function "Jump back to the start of the program" offers the following advantages:

- The programming of a jump mark at the start of the program is not necessary.
- The program restart can be controlled through the NC/PLC interface signal: DB21, ... DBX384.0 (control program branching)
• The timer for the program runtime can be reset to "0" at the restart of the program.
• The timer for workpiece counting can be incremented by "1" at program restart.

Application

The function is used, if the processing of subsequent workpieces is to be done through an automatic program restart e.g. in case of turning machine with bar loader/-changer.

Activation

The jump back takes place only when the following NC/PLC interface signal is set:
DB21, ... DBX384.0 (control program branching) = 1

If the signal is at "0", then no jump back is executed and the program processing is continued with the next part program block after the function call.

Parameter assignment

Program runtime

The runtime of the selected NC program is stored in the system variable $AC_CYCLE_TIME. When starting a new program, the system variable is automatically reset to "0" (see Chapter "Program runtime (Page 641)"

Via the following machine data it can be set that the system variable $AC_CYCLE_TIME is reset to "0" even in case of a program restart through the function "jump back to start of program":

MD27860 $MC_PROCESSTIMER_MODE (Activation of the program runtime measurement)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
<td>$AC_CYCLE_TIME is not reset to &quot;0&quot; by the function &quot;jump back to start of program&quot;.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$AC_CYCLE_TIME is reset to &quot;0&quot; by the function &quot;jump back to start of program&quot;.</td>
</tr>
</tbody>
</table>

Note

In order that the setting of bit 8 can become effective, the measurement of the current program runtime must be active (MD27860 bit 1 = 1).

Workpiece count

After the part program end (M02 / M30) has been attained, the activated workpiece counters ($AC_TOTAL_PARTS / $AC_ACTUAL_PARTS / $AC_SPECIAL_PARTS) are incremented by "1" (see Chapter "Workpiece Counter (Page 647)").

Via the following machine data it can be set that the activated workpiece counter is incremented even in case of a program restart through the function "jump back to start of program":
MD27880 $MC_PART_COUNTER (activation of workpiece counters)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td>$AC_TOTAL_PARTS is not incremented.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$AC_TOTAL_PARTS is incremented.</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>$AC_ACTUAL_PARTS is not incremented.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$AC_ACTUAL_PARTS is incremented.</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>$AC_SPECIAL_PARTS is not incremented.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$AC_SPECIAL_PARTS is incremented.</td>
</tr>
</tbody>
</table>

Programming

The function is called in the main or the part program via the command `GOTOS`.

Syntax: `GOTOS`

Parameters: none

Application in synchronized actions: not possible

Note

`GOTOS` internally initiates a `STOPRE` (pre-processing stop).

Example

<table>
<thead>
<tr>
<th>Programming</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 ...</td>
<td>; Beginning of the program</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N90 GOTOS</td>
<td>; Jump to beginning of the program</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

9.9.11 Program section repetitions

9.9.11.1 Overview

Function

The program section repetition allows the repetition of any labeled section of a part program.
Definition options of part program sections

The program repetition offers various options for defining a part program section that is supposed to be repeated:

- A single part program block
- A part program section after a start label
- A part program section between a start label and end label
- A part program section between a start label and the key word: ENDLABEL

References:
Programming Manual, Job Planning; Section "Program coordination"

9.9.11.2 Individual part program block

Functionality

Via REPEATB (B=Block) in part program block N150, the part program processing branches to the part program block N120 that is labeled START_1. This is repeated x number of times. If P is not specified, the program section is repeated exactly once. After the last repetition, the part program is continued with the part program block N160 following the REPEATB instruction.

```plaintext
N100 ... 
N120 START_1: ... ; Label: START_1 
N130 ... 
N140 ... 
N150 REPEATB START_1 P=n ; Repetition after: START_1 
N160 ... 
: 
```

Note

Label search direction

The part program block identified by the label can appear before or after the REPEATB statement. The search initially commences toward the start of the program. If the label is not found, a search is made in the direction of the program end.
### Programming

**Syntax:**

```
REPEATB <Label> [P=n]
```

- **Label**: Start label to which the instruction: REPEAT branches
  - Type: String
- **P**: Number of repetitions
  - Type: Integer

---

### 9.9.11.3 A part program section after a start label

**Functionality**

Via **REPEAT** in part program block N150, the part program processing branches to the part program block N120 that is labeled **START_1**. This part program block and all of the following part program blocks (N130 and N140) are repeated x number of times up to the part program block that contains the **REPEAT** instruction (N150). If P is not specified, the part program section (N120 - N140) is repeated exactly once. After the last repetition, the part program is continued with the part program block N160 following the **REPEAT** instruction.

```
: 
N100 ... 
N120 START_1: ... ; Start label: START_1 
N130 ... 
N140 ... 
N150 REPEAT START_1 P=n ; Repetition after: START_1 
N160 ... 
: 
```

---

### Note

**Label search direction**

The part program block marked with the Start label **must** come before the **REPEAT** instruction.
9.9.11.4 A part program section between a start label and end label

Functionality

Via **REPEAT** in part program block N160, the part program processing branches to the part program block N120 that is labeled **START_1** with a start label. This part program block and all the part program blocks up to and including the part program block marked with the end label **END_1** (N140) are repeated x number of times. If P is not specified, the part program section (N120 - N140) is repeated exactly once. After the last repetition, the part program is continued with the part program block N170 following the **REPEAT** instruction.

```
N100 ...  
N120 START_1: ... ; Start label: START_1 
N130 ...  
N140 END_1 ... ; End label: END_1 
N150 ...  
N160 REPEAT START_1 END_1 P=n ; Repetition: START_1 until END_1 
N170 ...  
```

Note

**Label search direction**

The program section marked with the Start and End labels can come before or after the **REPEAT** instruction. The search initially commences toward the start of the program. If the Start label is not found, a search is made in the direction of the program end.

If the **REPEAT** instruction is between the Start and End label, only the part program section from the Start label to the **REPEAT** instruction is repeated.

**REPEAT** instruction repeated.

Programming

**Syntax:**

```
REPEAT <Start_Label> <End_Label> [P=n]
```

- **Start_Label**: Start label to which the instruction: **REPEAT** branches.
  Beginning of the part program section that is repeated.
  Type: String

- **End_Label**: End of the part program section that is repeated.
  Type: String

- **P**: Number of repetitions
  Type: String

- **n**: Number of repetitions
  Type: Integer
9.9.11.5 A part program section between a Start label and the key word: ENDLABEL

Functionality

Via REPEAT in part program block N150, the part program processing branches to the part program block N120 that is labeled START_1 with a start label. This part program block and all the part program blocks up to and including the part program block marked with the key word ENDLABEL (N140) are repeated x number of times. If P is not specified, the part program section (N120 - N140) is repeated exactly once. After the last repetition, the part program is continued with the part program block N170 following the REPEAT instruction.

```
N100 ...
N120 START_1: ... ; Start label: START_1
N130 ...
N140 ENDLABEL: ... ; End label: Keyword ENDLABEL
N150 ...
N160 REPEAT START_1 END_1 P=n ; Repetition: START_1 until END_1
N170 ...
```

Note

Label search direction

The program section marked with the Start and End labels can come before or after the REPEAT instruction. The search initially commences toward the start of the program. If the Start label is not found, a search is made in the direction of the program end.

If no keyword ENDLABEL is located between the Start label and the REPEAT instruction, the part program section from the Start label to the REPEAT instruction is repeated.

Programming

Syntax: \( \text{REPEAT } <\text{Label}> \ [P=n] \)

- \(<\text{Label}>\): Start label to which the instruction: REPEAT branches. Beginning of the part program section that is repeated. Type: String
- \(P\): Number of repetitions
- \(n\): Number of repetitions Type: Integer
9.9.12 Event-driven program call (PROG_EVENT)

9.9.12.1 Function

Function

The "Event-driven program calls" function provides the option that when a certain event occurs in the NC, an NC program with user-defined content can be executed. For example, to initiate the basic setting of functions or initialize system and user variables.

Events

Triggering events can be:

- Program start
- End of program
- Channel reset
- Powering-up of the NC

The triggering events are selected with the machine data MD20108 $MC_PROG_EVENT_MASK (see Section "Parameterization (Page 549)").

User program

In the default setting the program _N_PROG_EVENT_SPF is activated after the triggering event occurs. Using machine data MD11620 $MN_PROG_EVENT_NAME, another program can be parameterized (see Chapter "Parameterization (Page 549)").

The activated user program is executed in the channel in which the event had occurred.

The application program is executed with the lowest priority and so can be interrupted by the user ASUB.

Processing sequence

Sequence when activated by an event: Program start

Initial state:

Channel: Reset status
Mode: AUTOMATIC (optional: Overstore (active)
MDI
TEACH IN

1. A program starts in the channel
2. Initialization sequence with evaluation of:
   - MD20112 $MC_START_MODE_MASK
3. Implicit call of _N_PROG_EVENT_SPF as part program
4. Processing of the data part of the main program
5. Processing of the program part of the main program

**Sequence when activated by an event: End of program**

Initial state

Channel: Active
Mode: AUTOMATIC (optional: Overstore (active)
MDI
TEACH IN

1. In the channel, the end of program block is exchanged in the executed program.
2. End of program is executed, evaluation of the following machine data:
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE
3. Implicit call of the parameterized user program as ASUB
4. A reset is executed in the channel, evaluation of the following machine data:
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE

**Sequence when activated by an event: Channel reset**

Initial state:

Channel: Any
Mode: Any

1. Control activates reset-sequence with evaluation of machine data:
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE
2. Implicit call of the parameterized user program as ASUB
3. Control activates reset-sequence with evaluation of machine data:
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE
Sequence when activated by an event: Powering-up of the NC

1. Control activates after power-up reset-sequence with evaluation of machine data:
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE

2. Implicit call of the parameterized user program as ASUB

3. Control activates reset-sequence with evaluation of machine data:
   - MD20110 $MC_RESET_MODE_MASK
   - MD20150 $MC_GCODE_RESET_VALUES
   - MD20152 $MC_GCODE_RESET_MODE

NC/PLC interface signals Program and channel state

The following diagrams show that the signal sequence for the interface signals "Program state" and "Channel state" during the event-driven program call:

![Signal sequence diagram](image)

Figure 9-7 Signal sequence when activated by program start and program end
**9.9 Program operation**

**Interface signals DB21, ... DBX35.4 (program state interrupted) and DB21, ... DBX35.7 (channel state reset)**

- The interface signals are only set if the activated, event-driven user program has already been exited.
- The interface signals are not set between:
  - end of program and the start of the event-driven user program
  - Channel reset and the start of the event-driven user program

**NC/PLC interface signals Initiating event**

The initiating event is displayed using interface signals DB21, ... DBX376.0 ... 4:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>Program start from channel state &quot;Reset&quot;</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>End of program</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Channel reset</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Powering-up of the NC</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1. Program start after block search (see &quot;Automatic Start of an ASUB after block search (Page 485)&quot;&quot;)</td>
</tr>
</tbody>
</table>

If the event-driven user program is exited, or canceled with a reset, then the interface signals are again deleted.

The interface signals are available as a minimum for the duration of a complete PLC cycle.
9.9.12.2 Parameterization

Events

The events that activate the user program are selected on a channel-specific basis using machine data:

MD20108 $MC_PROG_EVENT_MASK, bit x

Note

MD20108 $MC_PROG_EVENT_MASK is not evaluated in the simulation.

Program

The program (default setting: _N_PROG_EVENT_SPF) must be loaded and cleared.

Default setting

As default setting, when an event occurs, user program _N_CMA_DIR/_N_PROG_EVENT_SPF is executed.

User-specific setting

If, for an event, another user program is to be executed, the program name must be entered into the following machine data:

MD11620 $MN_PROG_EVENT_NAME

The program must be in a cycle directory.

The following search path is run through when an event occurs.

1. /_N_CUS_DIR/ (user cycles)
2. /_N_CMA_DIR/ (manufacturer cycles)
3. /_N_CST_DIR/ (standard cycles)

The first program found with the specified name is executed.

Note

- The specified name is checked syntactically as in case of a subprogram name, i.e. the first two characters must be letters or underscores (no numbers). Prefix (_N_) and suffix (_SPF) of the program names are added automatically, if not specified.
- The same protection mechanisms that can be activated for cycles (protection levels for writing, reading, etc.) are activated.

Behavior when starting a user ASUB

The behavior of the function "event-driven program call" upon start of a user ASUB from the reset channel state can be set channel-specific with the machine data:
Behavior when the single-block processing is set

The behavior of the function "event-driven program call" in case of set single-block processing can be set channel-specific with the machine data:

**MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>In the event-driven user program:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● After an activation through part program start:</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>The single-block processing is active</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>The single-block processing is suppressed</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>● After an activation by part program end:</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>The single-block processing is active</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>The single-block processing is suppressed</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>● After an activation by an operator panel reset:</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>The single-block processing is active</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>The single-block processing is suppressed</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>● After an activation by powering up:</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>The single-block processing is active</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>The single-block processing is suppressed</td>
</tr>
</tbody>
</table>

If the single-block processing is suppressed, then the event-driven user program is processed without interruption.

**Note**

- **MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK** affects all single-block processing types.
- The single-block processing in the event-driven user program can be switched-off through the following configuration:
  MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (prevent single-block stop) bit 0 = 1
  The differentiated settings in MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK are then ineffective.

Behavior when the read-in disable is set

The behavior of the function "event-driven program call" in case of set read-in disable (DB21, ... DBX6.1 = 1) can be set channel-specific with the machine data:
### MD20107 \$MC_PROG_EVENT_IGN_INHIBIT

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>In the event-driven user program:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• After an activation through part program start:</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>The read-in disable is effective</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the user program is exited with the RET part program command, then an executable block is not created</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>The read-in disable is suppressed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the user program is exited with the RET part program command, then an executable block analog to M17) is created.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>The read-in disable is effective</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>The read-in disable is suppressed</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>The read-in disable is effective</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>The read-in disable is suppressed</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>The read-in disable is effective</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>The read-in disable is suppressed</td>
</tr>
</tbody>
</table>

#### Suppress updating of the display of the program and channel states.

In order to avoid a flickering of the display of the program and the channel states in the operator panel, the updating of the display can be suppressed for the execution of the normally very brief event-driven user program. In the display, the program and channel states remain visible before activation of the event-driven user program (e.g. "program state canceled" and "channel state reset").

The parameterization of this function is performed with the channel-specific machine data:

### MD20192 \$MC_PROG_EVENT_IGN_PROG_STATE

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>During execution of an event-driven user program, the updating of the display of the program and channel states is:</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Not suppressed</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Suppressed</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>Not suppressed</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Suppressed</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>Not suppressed</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Suppressed</td>
</tr>
</tbody>
</table>
Note

The system variables $AC_STAT and $AC_PROG are not affected by this function, i.e. in the running event-driven user program, $AC_STAT is set to "active" and $AC_PROG to "running". NC/PLC interface signals DB21, ... DBX35.0-7 ("Program state ..." and "Channel state ...") also remain unaffected.

Behavior for NC Stop

The behavior of the function "event-driven program call" for NC Stop (i.e. NC/PLC interface signal DB21, ... DBX7.2, 7.3 or 7.4 is set) can be set channel-specific for the triggering event part program end, operator panel reset and run-up with the machine data:

MD20193 $MC_PROG_EVENT_IGN_STOP

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>● After an activation through part program end:</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Stopped/prevented at NC stop</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Despite NC Stop been completely processed</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>● After an activation through operator panel reset:</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Stopped/prevented at NC stop</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Despite NC Stop been completely processed</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>● After an activation through run-up:</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Stopped/prevented at NC stop</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Despite NC Stop been completely processed</td>
</tr>
</tbody>
</table>

In this way, an edge change of the interface signal DB21, ... DBX7.3 (NC Stop) initiated by the user by activating the NC Stop key in case of reset or power-up is ignored during the execution of the event-driven user program and an undesired stop behavior at the machine is prevented.

Note

A programming of DELAYFSTON/DELAYFSTOP in the event-driven user program cannot be provided with the behavior set with MD20193, because the NC Stop can cause an interruption before the execution of the first command DELAYFSTON.

9.9.12.3 Programming

User program

End of program
The following must be kept in mind, if the user program is to be activated through the part program start.

- The user program must be ended with M17 or RET.
- A jump back by means of REPOS command is not permitted and leads to an alarm.

**Block display**

The display can be suppressed in the current block display using the DISPLOF attribute in the PROC statement.

**Processing status**

Via the user M function the PLC can be informed about the processing status of the event-driven user program.

### Scan for triggering event

The event, which causes the activation of the user program, can be queried in the user program with the following system variable:

$P_PROG_EVENT (event-driven program call active)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Activation through part program start</td>
</tr>
<tr>
<td>2</td>
<td>Activation through part program end</td>
</tr>
<tr>
<td>3</td>
<td>Activation through Operator panel reset</td>
</tr>
<tr>
<td>4</td>
<td>Activation through Power up</td>
</tr>
<tr>
<td>5</td>
<td>Activation after output of the last action block after Block search (see &quot;Automatic Start of an ASUB after block search (Page 485)&quot;)</td>
</tr>
</tbody>
</table>

### Query of the current channel

The application program is basically processed in the channel, in which the corresponding event has occurred. The current channel is queried in the user program with the following system variables:

$P_CHANNO (query of the current channel number)

**Note**

Power up is an event that takes place in all channels.
9.9.12.4 Boundary conditions

Emergency stop / error message

If an error is present when the operator panel is reset or after powerup EMERGENCY STOP or Mode group/NCKContinue, then the event-driven user program will only be processed after EMERGENCY STOP or the error has been acknowledged in all channels.

Note

The power up event occurs in all channels at the same time.

9.9.12.5 Examples

Example 1: Call of all events set with MD20108

Parameter assignment:

MD20108 $MC_PROG_EVENT_MASK = 'H0F'

Call of _N_PROG_EVENT_SPF for:
- Part program start
- Part program end
- Operator panel reset
- Ramp-Up

Programming:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC PROG_EVENT DISPLOF</td>
<td></td>
</tr>
<tr>
<td>IF ($P_PROG_EVENT==1)</td>
<td>; Processing for part program start.</td>
</tr>
<tr>
<td>MY_GUD_VAR=0</td>
<td>; Initialize GUD variable</td>
</tr>
<tr>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>IF ($P_PROG_EVENT==2) OR ($P_PROG_EVENT==3)</td>
<td>; Processing for part program end and operator panel reset.</td>
</tr>
<tr>
<td>DRFOF</td>
<td>; Deactivate DRF offsets</td>
</tr>
<tr>
<td>IF $MC_CHAN_NAME=&quot;CHAN1&quot;</td>
<td></td>
</tr>
<tr>
<td>CANCEL(2)</td>
<td>; Delete modal synchronized action 2</td>
</tr>
<tr>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>IF ($P_PROG_EVENT==4)</td>
<td>; Sequence for power-up</td>
</tr>
</tbody>
</table>
Example 2: Call through Operator panel reset

Parameter assignment:

MD20108 $MC_PROG_EVENT_MASK = 'H04'  
Call of _N_PROG_EVENT_SPF for:
- Operator panel reset

Programming:

```
PROC PROG_EVENT DISPLOF
  N10 DRFOF ; Deactivate DRF offsets
  N20 M17
```

Example 3: Initialization of the function

Section of the startup file (_N_INITIAL_INI):

```
...  
  CHANDATA(3) ; Initialization for channel 3
  $MC_PROG_EVENT_IGN_INHIBIT='H04F'
  $MC_PROG_EVENT_MASK='H04'
  ...
```

Meaning:
The part program _N_PROG_EVENT_SPF from the directory _N_CMA_DIR should be started automatically with the RESET key and processed till the end, regardless of whether the read-in disable is activated or deactivated.
9.9.13 Influencing the Stop events through Stop delay area

Stop delay area

The reaction to a stop event can be influenced by the conditioned interruptible area in the current part program. Such a program area is called stop delay area.

Within the stop delay areas there should be no stop and the feed should not be changed. Stops do not take effect until the program section has been completed (Example: Making of a thread).

A stop delay area is defined with the part program commands:

- DELAYFSTON Start of a stop delay area
- DELAYFSTOF End of a stop delay area

References:
Programming Manual, Job Planning

Stop events

Overview of the NCK events that cause a stop:

<table>
<thead>
<tr>
<th>NCK events</th>
<th>Reaction</th>
<th>Stop criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset and mode group RESET</td>
<td>Immediate</td>
<td>IS: DB21, ... DBX7.7 and DB11 DBX20.7</td>
</tr>
<tr>
<td>PROG_END</td>
<td>Alarm 16954</td>
<td>NC prog.: M30</td>
</tr>
<tr>
<td>Interrupt</td>
<td>Delayed</td>
<td>IS: &quot;FC-9&quot; and ASUB DB10 DBB1</td>
</tr>
<tr>
<td>DELDISTOGO_SYNC</td>
<td>Immediate</td>
<td>IS: &quot;Delete distance-to-go&quot; DB21, ... DBX6.2 and axial</td>
</tr>
<tr>
<td>PROGRESETREPEAT</td>
<td>Delayed</td>
<td>IS: &quot;Clear number of subprogram passes&quot; DB21, ... DBX6.3</td>
</tr>
<tr>
<td>PROGCANCELSUB</td>
<td>Delayed</td>
<td>IS: &quot;Program level abort&quot; DB21, ... DBX6.4</td>
</tr>
</tbody>
</table>
| SINGLEBLOCKSTOP            | Delayed  | In the stop delay area: NC stops at the end of the 1st block outside the stop delay area.  
                            |          | Single block is active before the stop delay area: 
                            |          | IS: "NC Stop at block limit" DB21, ... DBX7.2  |
| SINGLEBLOCK_IPO            | Delayed  | IS: "Activate single-block type 1" DB11 DBX21.7                            |
| SINGLEBLOCK_DECODIER        | Delayed  | IS: "Activate single-block type 2" DB11 DBX21.6                            |
| STOPALL                    | Immediate| IS: DB21, ... DBX7.4 and DB11 DBX20.6                                      |
| STOPPROG                   | Delayed  | IS: DB21, ... DBX7.3 and DB11 DBX20.5                                      |
| OVERSTORE_BUFFER_END_REACHED| Alarm 16954| NC prog.: Stop because of empty overstore buffer                          |
| PREP_STOP                  | Alarm 16954| NC prog.: STOPRE and all implicit Stopres                                  |
| PROG_STOP                  | Alarm 16954| NC prog.: M0 and M1                                                       |
| STOPPROGATBLOCKEND         | Delayed  | IS: "NC Stop at block limit" DB21, ... DBX7.2                             |
| STOPPROGATSUPEND           | System fault| Subprogram end should always deselect the stop delay area.  |
| WAITM                      | Alarm 16954| NC prog.: WAITM                                                            |
| WAITE                      | Alarm 16954| NC prog.: WAITE                                                           |
| INIT_SYNC                  | Alarm 16954| NC prog.: INIT with parameter "S"                                           |
| MMCCMD                     | Alarm 16954| NC prog.: MMC( STRING, CHAR )                                              |
### Reaction

The reaction to a stop event can be:

- **Immediate**
  - Stops immediately even in the stop delay area. Is known as a "hard stop event".

- **Delayed**
  - Does not stop (even short-term) until after the stop delay area. Is known as a "soft stop event".

- **Alarm 16954**
  - Program is aborted because illegal program commands have been used in the stop delay area.

- **Alarm 16955**
  - Program is continued, an illegal action has taken place in the stop delay area.

- **Alarm 16957**
  - The program area (stop delay area) enclosed by \texttt{DELAYFSTON} and \texttt{DELAYFSTOF} could not be activated. As a result, every stop will take effect immediately and is not subject to a delay! This will always occur when the deceleration begins before the stop delay area but ends within the stop delay area. Likewise, if the stop delay area is entered with an override of 0, the stop delay area also cannot be activated. (Example: A \texttt{G4} before the stop delay area allows the user to reduce the override to 0. The next block in the stop delay area then begins with override 0 and the described alarm situation occurs.)

### Note

\texttt{MD11411 $MN_ENABLE_ALARM_MASK} (activation of warnings) Bit 7 activates this alarm.

### Stop criteria
A stop event can be triggered by the following:

- NC/PLC interface signals from the PLC → "Hard" stop event
- Alarms with NOREADY response → "Hard" stop event
- Stop key → "Soft" stop event
- Read-in disable → "Soft" stop event
- Single block → "Soft" stop event

**Note**

Some NCK events are stopped for a short time, in order to perform a switching operation, and restart immediately. These include, e.g. the ASUB that stops the contour briefly in order to then start the ASUB immediately. These events are also allowed in the stop delay area, however they are pushed back to its end and are thus considered "soft stop events".

**Conditions**

The following conditions apply while a stop delay area is being processed:

- A change in the feed is ignored while in the stop delay area. A feed disable is thus not effective until the program area has been exited, and is stopped.
- None of the main run axes, such as command axes and positioning axes, which are traversed with POSA, are stopped.
- Part program command G4 is permitted in the stop delay area. Other part program commands that cause a stop in the meantime (e.g. WAITM) are not permitted and trigger the alarm 16954.
- A stop delay area entered with an override of 0% will not be accepted!

**9.10 Asynchronous subprograms (ASUBs), interrupt routines**

**9.10.1 Function**

**9.10.1.1 General functionality**

**Note**

The terms "asynchronous subprogram (ASUB)" and "interrupt routine" are used interchangeably in the description below to refer to the same functionality.
Interrupt routines

Interrupt routines are NC programs which are started by interrupt events (interrupt inputs, process or machine status).

A part program block currently being executed will be aborted by the activation of an interrupt routine if it is not specifically declared to be locked against interruption. It is possible to continue the part program at the point of interruption at a later stage.

Definition of interrupt routines

The command `SETINT` or an interrupt signal via the PI service "ASUB" must be assigned to a part program, which is supposed to act as interrupt routine. This turns the part program into an interrupt routine.

Interrupt signals

- A total of eight interrupt signals (inputs) are available.
- All inputs can be controlled via the PLC.
- The first four interrupt signals are also controlled via the four fast NC inputs of the NCU module.
• The signal states of the fast NC inputs can be read out via the NC/PLC interface (DB10).
• The transmission of the fast NC inputs to the interrupt signals can be disabled via the NC/PLC interface (DB10).

![Diagram of NC inputs to interrupt signals](image)

Figure 9-9 Interrupt signals

For further information about PLC control of the rapid NC inputs (interrupt signals) see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 821)".

References:
Function Manual, Extended Function; Digital and Analog NCK I/O (A4)

Call of interrupt routines

**During program mode**
During program mode, i.e. in the AUTOMATIC or MDA mode, interrupt routines can always be called.

**Outside the program mode**
Outside the program mode, interrupt routines can be called in the following modes, machine functions and states:
• JOG, JOG REF
• MDA Teach In, MDA Teach In REF, MDA Teach In JOG, MDA REF, MDA JOG
• AUTOMATIC, program states "stopped", "ready"
• Axis state "Not referenced"

If an interrupt routine is started during JOG or JOG REF, the current action is aborted.
Activation

The activation of an interrupt routine is initiated via:

- 0/1 edge of the interrupt signal, triggered by a 0/1 edge at the associated fast NC input
- Call of the "Function call ASUB" (see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 821)"
- Setting an output via synchronized action which is parameterized on an interrupt via short-circuit (see "Examples (Page 570)"

References:
Function Manual, Synchronized Actions

Display

The activation of an interrupt routine is shown channel-specifically with the following NC/PLC interface signal:

DB21, … DBX378.0 == 1 (ASUB active)

9.10.1.2 Sequence of an interrupt routine in program operation

Decelerating the axes

Upon activation, all machine axes are decelerated to a standstill according to the acceleration ramp (MD32300 $MA_MAX AX ACCEL), and the axis positions are stored.

Reorganization

In addition to decelerating the axes, the previously decoded calculation blocks are calculated back to the interruption block, i.e. all the variables, frames and G codes are assigned the value that they would have at the point of interruption if the part program had not been previously decoded. These values are transferred to the buffer so that they can be called up again when the interrupt routine is completed.

Exceptions where no reorganization is possible:

- In thread cutting blocks
- With complex geometries (e.g. spline or radius compensation)

Processing of interrupt routine

The interrupt routine is automatically started on completion of reorganization.

The system handles the interrupt routine like a normal part program (nesting depth, etc.)

End of interrupt routine

After the end identifier (M02, M30, M17) of the interrupt routine has been processed, the axis traverses by default to the end position programmed in the part program block following the interruption block.
A `REPOS` instruction must have been programmed at the end of the interrupt routine if return positioning to the point of interruption is required, e.g.

```
N104 REPOS L M17
```

9.10.1.3 **Interrupt routine with REPOSA**

If an interrupt routine with `REPOSA` triggered by the PLC (block FC9) "interrupted" in channel status in program operation is completed, then the following sequence is typical:

1. Before the re-approach to the contour, the controller stops and goes to program status "Stopped". The following NC/PLC-interface signal is set: DB21, ... DBX318.0 (ASUB is stopped)
2. The operator presses the START key. Thereupon, the signal DB21, ... DBX318.0 is reset and the re-approach motion starts.
3. At the end of the re-approach motion, the FC9 signal "ASUB done" is set and the path of the interrupted part program is continued.

**Note**

The NC/PLC-interface signal DB21, ... DBX318.0 (ASUB is stopped) is available only for the following case: Interrupt "interrupted" in program operation in the channel status.

**Note**

In case of interrupt routines that close without REPOS, the signals "Asub-Done" and DB21, ... DBX318.0 (ASUB is stopped) occur at the same time.
### 9.10.1.4 NC response

The different reactions of the control to an activated interrupt routine in the various operating modes are given in the following table:

<table>
<thead>
<tr>
<th>Status of NC</th>
<th>ASUB start</th>
<th>Control system reaction</th>
</tr>
</thead>
</table>
| Program is active | Interrupt, (PLC) | 1. Fast retraction or stop axes  
2. Interrupt the program for the duration of the ASUB  
3. Approach of the interruption point, if REPOS in ASUB  
4. Continuation of the part program |
| Reset | Interrupt, (PLC) | The ASUB is executed like a main program. Reset (without M30) is executed at the end of the ASUB. The next control system status depends on the following machine data:  
MD20110 $MC_RESET_MODE_MASK  
MD20112 $MC_START_MODE_MASK |
| Program operation (AUTO-MATC or MDA) + channel stopped | Interrupt, (PLC) | ASUB is executed. At the end of the ASUB the STOP state is reapplied. If REPOS in the ASUB:  
• The ASUB processing is stopped before the approach block.  
• The approach movement can be initiated with the Start key. |
| Manual mode + channel stopped | Interrupt, (PLC) | Control system assumes the status “internal program execution mode” for the addressed channel (not evident externally) and then activates the ASUB. The selected operating mode remains valid. The original status is resumed after execution of the ASUB (M17). |
| JOG AUTO Teach-In AUTO Teach reference point. | Interrupt, (PLC) | Stop processing, evaluate:  
MD11602 $MN_ASUP_START_MASK  
MD11604 $MN_ASUP_START_PRIO_LEVEL  
Internal switchover to “internal program execution mode” if appropriate, activate ASUB, restore status prior to ASUB start. Any LIFTFAST defined with SETINT is not activated in JOG mode. |
| MDA JOG, MDA Teach-In, MDA Teach reference point. | Interrupt, (PLC) | The current active motion is stopped. The distance-to-go is deleted. The remaining sequence of operations is the same as for “Manual mode, channel stopped”. |
| Manual mode + channel running | Interrupt, (PLC) | When the following machine data is set, the user ASUB is processed despite the “NC Start disable” alarm response:  
MD20194 $MC_IGNORE_NONCSTART_ASUP |
| User alarm 65500 - 65999, channel in reset | Interrupt, (PLC) | |

References:
Function Manual Basic Functions; Axes, Coordinate Systems, Frames (K2), Chapter: “Workpiece-related actual-value system”
9.10.2 Parameterization

Effect of mode group-specific signals

The effect of the mode group-specific signals (NC/PLC interface signals DB11) on channels of the mode group that is currently processing the interrupt routines, is set in the machine data:

\[
\text{MD11600 $\text{SMN\_BAG\_MASK}$.bit n = <value>}
\]

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>The mode group-specific signals (NC/PLC interface signals DB11) are effective.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>The mode group-specific signals (NC/PLC interface signals DB11) are not effective.</td>
</tr>
</tbody>
</table>

MD11600 also controls whether the internal program execution mode is imported only for the channel in which the interrupt routine has been activated, or to all channels in the mode group.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>A mode group switchover is performed in all channels of the mode group.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>An internal mode group switchover is only performed in the channel in which an interrupt routine is active (only possible with bit 0==1).</td>
</tr>
</tbody>
</table>

If, because of the machine data setting, the channel in which the interrupt is being processed has left the mode group, the mode group signals "Mode group reset", "Mode group stop", etc. have no effect on this channel. In this way, the interrupt routine is run without being disturbed by the mode group signals.

Parameterizable start enable

In the basic setting, the start of an ASUB is blocked by the following states in the channel:

- Stop through NC stop, M0 or M01
- Not all axes in the channel are referenced
- Read-in disable is active (DB21, ... DBX6.1 = 1)

The start can be enabled for specific states via the following machine data.
NC-specific start enable for NC stop, M0, M01, read-in disable
MD11602 $MN_ASUP_START_MASK

Channel-specific start enable for non-referenced axes in the channel
- The start enable can be set separately for the following states via the machine data for event-driven program calls (ProgEvent) for non-referenced axes in the channel:
  - Part program start from channel state "Reset"
  - Part program end
  - Reset
  - Power On
  - First start after search
MD20105 $MC_PROG_EVENT_IGN_REFP_LOCK, Bit<n> = TRUE
n: For all required event-driven program calls (prog events)

- The start enable can be issued interrupt-specifically via the machine data for the ASUB with non-referenced axes in the channel:
MD20115 $MC_IGNORE_REFP_LOCK_ASUP, bit<n> = TRUE
n: For all required user interrupts

### NOTICE

**System interrupts**

Interrupts 9 and 10 used by the system are released with MD20115 $MC_IGNORE_REFP_LOCK_ASUP, bit 8 and bit 9. Interrupt 9 starts an ASUB which contains traversing motions.

---

**Manual start enable**

If an ASUB cannot be started automatically because of the parameterized start enables, the user can manually issue a start enable by triggering **NC start**

**Note**

The ASUB for "fast retraction from the contour" (LIFTFAST) is started in every case.

---

**Manual traversing during an ASUB interruption in JOG mode**

Axes can be manually traversed by the operator using the traversing keys during the interruption of an ASUB started automatically in JOG mode. The function is enabled via:
K1: Mode group, channel, program operation, reset response

9.10 Asynchronous subprograms (ASUBs), interrupt routines

MD11602 $MN_ASUP_START_MASK.bit 3

---

Note

Multi-channel systems

In multi-channel systems, the following machine data must also be set:

MD11600 $MN_BAG_MASK, bit 1 = 1

---

Continuation of the ASUB

After manually traversing the axes, NC Start must be initiated by the operator. The axes are automatically traversed to the interruption point (REPOS). The ASUB is then continued at the interruption point.

Application example

In case of a single-slide turning machine a stock removal cycle is started as ASUB in the JOG mode and with this a shaft several meters long is machined. During machining it is necessary to change the cutting edge of the tool. The machine operator stops the ASUB and retracts the axes manually using the traversing keys in order to change the cutting edge of the tool. After changing the cutting edge, the operator activates NC Start. Repositioning at the interruption point is performed with the REPOS operation. Thereafter, the ASUB is continued.

Supplementary condition

Manual traversing of axes during the interruption of an ASUB in JOG mode is only possible if the ASUB has been activated from the program state "Aborted", channel state "Reset".

Effectiveness of the parameterized start enables

The user-specific interrupt signals are assigned the priorities 1 - 8, with 1 = highest priority (see Section "Programming (Page 568)"). Starting from the highest, up to which priority the parameterized start enables apply for the associated ASUB is defined with the following machine data:

MD11604 $MN_ASUP_START_PRIO_LEVEL = <priority>

Behavior with read-in disable

The following channel-specific machine data is used to set for each interrupt signal whether the assigned ASUB is processed despite a set read-in disable (DB21, ... DBX6.1 = 1), or whether the read-in disable is to apply:

MD20116 $MC_IGNORE_INHIBIT_ASUP, bit 0 - bit 31

Bit x is assigned to the interrupt signal (x+1).

Supplementary conditions

The settings in MD20116 $MC_IGNORE_INHIBIT_ASUP have no effect if the read-in disable in the ASUB is always to be ignored:

MD11602 $MN_ASUP_START_MASK, bit 2 = 1
Behavior when the single-block processing is set

The following channel-specific machine data is used to set for each interrupt signal whether the assigned ASUBs are processed without interruption with active single-block processing or whether the single-block processing is to apply:

\[ \text{MD20117 $MC\_IGNORE\_SINGLEBLOCK\_ASUP$, bit 0 - bit 31} \]

Bit x is assigned to the interrupt signal (x+1).

Supplementary conditions

- MD20117 $MC\_IGNORE\_SINGLEBLOCK\_ASUP$ only applies for IPO single block (SBL1).
- The settings in MD20117 $MC\_IGNORE\_SINGLEBLOCK\_ASUP$ have no effect if the single-block processing in the ASUB is always to be ignored:
  \[ \text{MD10702 $MN\_IGNORE\_SINGLEBLOCK\_MASK$, bit 1 = 1} \]

Suppressing the updating of the "Program and channel state" display

To avoid a flickering of the display of the program and the channel states on the user interface when executing a very short ASUB, the updating of the display can be suppressed. The program and channel state before activation of the ASUB is then displayed. The parameter assignment is performed via channel-specific machine data:

\[ \text{MD20191 $MC\_IGN\_PROG\_STATE\_ASUP$, bit 0 - bit 31} \]

Bit x is assigned to the interrupt signal (x+1).

The following NC/PLC interface signal is set when executing an ASUB with suppressed display:

\[ \text{DB21, … DBX378.1 == 1} \text{ (still ASUB active)} \]

Special system variables and NC/PLC interface signals

The system variables and NC/PLC interface signals for the program state and the channel state are not affected by the suppression of the display during the execution of an ASUB:

- $\text{AC\_STAT}$ (channel state)
- $\text{AC\_PROG}$ (program state)
- DB21, … DBX35.0 - 4 (program state)
- DB21, … DBX35.5 - 7 (channel state)

Fast retraction from contour (LIFTFAST)

The following machine data can be used to set whether the retraction direction for "Fast retraction from the contour" (LIFTFAST) is to be mirrored when the "Mirror" function is active:

\[ \text{MD21202 $MC\_LIFTFAST\_WITH\_MIRROR$ (fast retraction with mirroring)} \]

The mirroring of the retraction direction only refers to the direction components perpendicular to the tool direction.

References

A detailed description of the machine data can be found in:

K1: Mode group, channel, program operation, reset response

9.10 Asynchronous subprograms (ASUBs), interrupt routines
9.10.3 Programming

Assignment interrupt signal ↔ part program

The assignment interrupt signal ↔ part program is performed with the command SETINT.

Example:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N20 SETINT(3) ABHEBEN_Z</td>
<td>; If input 3 switches, then interrupt routine &quot;ABHEBEN_Z&quot; should start.</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Together with SETINT additionally the following commands can be programmed:

- **LIFTFAST**
  
  When the interrupt signal arrives, a "Fast retraction of the tool from the contour" is executed before the interrupt routine starts. The motion direction for the fast retraction is specified by the program instruction ALF.

- **BLSYNC**

  Upon receiving the interrupt signal, the current program block is processed and only then is the interrupt routine started.

**Note**

The assignment interrupt signal ↔ part program is cleared when the following happens:

- Channel in Reset state
- **CLRINT** instruction in part program

**Priorities**

If several SETINT instructions are in the part program and therefore several signals can be simultaneously received, the assigned interrupt routines must be allocated priorities that define the sequence in which the interrupt routines are executed:

`PRIO=<value>`

There are priorities from 1 to 128. Priority 1 corresponds to the highest priority.

Example:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N20 SETINT(3) PRIO=2 ABHEBEN_Z</td>
<td>; If input 3 switches, then interrupt routine &quot;ABHEBEN_Z&quot; should start.</td>
</tr>
<tr>
<td>N30 SETINT(2) PRIO=2 ABHEBEN_X</td>
<td>; If input 2 switches, then interrupt routine &quot;ABHEBEN_X&quot; should start.</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
The interrupt routines are executed in the sequence of the priority values if the inputs become available simultaneously (are energized simultaneously): First "ABHEBEN_Z", then "ABHEBEN_X".

REPOS-query

With interrupt routines, sequences may be generated for which there is no unambiguous return to an interruption point in the block processing sequence (REPOS).

The system variable $P_REPINF can be used to scan the ASUB to determine whether a REPOS is possible.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Repositioning with REPOS not possible because:</td>
</tr>
<tr>
<td></td>
<td>● Not called in the ASUB</td>
</tr>
<tr>
<td></td>
<td>● ASUB ran from reset status</td>
</tr>
<tr>
<td></td>
<td>● ASUB ran from JOG</td>
</tr>
<tr>
<td>1</td>
<td>Repositioning with REPOS possible in ASUB</td>
</tr>
</tbody>
</table>

Determining the cause of activation

With the system variable $AC_ASUP the cause leading to the activation of an interrupt routine is specified bit-coded and can be read in the part program and in synchronized actions (see "User-specific ASUB for RET and REPOS > Programming (Page 573)").

Flexible programming

The following commands help in the flexible programming of interrupt routines:

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAVE</td>
<td>If the SAVE command has been used to define the interrupt routine, the G codes, frames and transformations previously active in the interrupted part program become operative again as soon as the interrupt routine is ended.</td>
</tr>
<tr>
<td>DISABLE</td>
<td>The DISABLE command can be set to protect part program sections from being interrupted by an interrupt routine. The assignment interrupt signal ↔ part program is maintained but the interrupt routine no longer responds to the 0/1 signal transition.</td>
</tr>
<tr>
<td>ENABLE</td>
<td>The DISABLE command can be reset with the ENABLE command. Interrupt routines are not activated until the next 0/1 transition of the interrupt signal.</td>
</tr>
<tr>
<td>CLRINT</td>
<td>Clear assignment interrupt signal ↔ part program</td>
</tr>
</tbody>
</table>

References

Programming Manual, Job Planning; Section: "Flexible NC Programming" > "Interrupt routine (ASUB)"

Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2
9.10.4 Restrictions

Cross-mode Start of interrupt routines

Settings to be checked
- MD11600 $MN_BAG_MASK
- MD11604 $MN_ASUP_START_PRIO_LEVEL
- Interrupt assignment priority

Recommended settings
NC-specific machine data:
- MD11600 $MN_BAG_MASK = 'H11'
- MD11602 $MN_ASUP_START_MASK = 'H101'
- MD11604 $MN_ASUP_START_PRIO_LEVEL = 7

Channel-specific machine data for the channel in which the ASUB is started or generally for all channels:
- MD20105 $MC_PROG_EVENT_IGN_REFP_LOCK, Bit<n> = TRUE
  n: For all required event-driven program calls (prog events)
- MD20115 $MC_IGNORE_REFP_LOCK_ASUP, bit<n> = TRUE
  n: For all required user interrupts

NOTICE
System interrupts
Interrupts 9 and 10 used by the system are released with MD20115 $MC_IGNORE_REFP_LOCK_ASUP, bit 8 and bit 9. Interrupt 9 starts an ASUB which contains traversing motions.

9.10.5 Examples

Activation of an interrupt routine via synchronous action
1. Parameterize two active digital I/O bytes:
   - MD10350 $MN_FASTIO_DIG_NUM_INPUTS = 2
   - MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS = 2
2. Parameterize a short-circuit with OR operation from output 9 to input 9:
   - Input 1, input byte 2 = (output 1, output byte 2) OR (HW input signal 1, input byte 2):
     MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT[0]="H0102B102"
3. Assign the HW input byte to the SETINT interrupt programming:
   - Input byte 2:
     MD21210 $MC_SETINT_ASSIGN_FASTIN = 2

4. Define input as ASUB trigger:
   - Input 1 in the second input byte, i.e. absolute input 9, starts SYNCASUP program
     SETINT(1) PRIO=1 SYNCASUP

5. Define a synchronized action to set the output:
   - Synchronized action with ID 1 always sets output 9 to 1 when the value of the
     standardized path parameter becomes >= 0.5:
     IDS=1 EVERY $$AC_PATHN >= 0.5 DO $A_OUT_[9]=1

9.11 User-specific ASUB for RET and REPOS

9.11.1 Function

Function

The NCK software supplied contains preprogrammed processes (internal ASUBs) for
implementation of the RET and REPOS functions. They can be replaced by user-specific
ASUBs written by the machine tool manufacturer.

⚠️ DANGER

Programming fault

The machine manufacturer is responsible for the contents of ASUB routines used to replace
ASUP.SYF supplied by Siemens.

Installation

In the manufacturer directory _N_CMA_DIR or in the user directory _N_CUS_DIR a routine
with the name "_N_ASUP_SPF" can be loaded. It must implement the actions desired by the
user for the functions RET and REPOS.

9.11.2 Parameter assignment

Activation

The parameters for the activation of the user-specific routine "_N_ASUP_SPF" are set with the
machine data:

MD11610 $MN_ASUP_EDITABLE (activation of a user-specific ASUB).
Bit 0 and bit 1 specify which of the internal system routines are to be replaced by the user-specific ASUB:

<table>
<thead>
<tr>
<th>Binary value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Neither in case of RET nor in case of REPOS is the user-specific routine _N_ASUP_SPF activated.</td>
</tr>
<tr>
<td>1</td>
<td>If the user-defined routine is activated for RET, then the routine provided in the system is activated for REPOS.</td>
</tr>
<tr>
<td>2</td>
<td>If the user-defined routine is activated for REPOS, then the routine provided in the system is activated for RET.</td>
</tr>
<tr>
<td>3</td>
<td>As in case RET and also in case of REPOS the user-specific routine is activated.</td>
</tr>
</tbody>
</table>

Bit 2 defines in which directory the user-specific routine is to be searched first in case of activation.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>The user-specific routine is searched first in the user directory _N_CUS_DIR.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>The user-specific routine is searched first in the manufacturer directory _N_CMA_DIR.</td>
</tr>
</tbody>
</table>

**Defining a level of protection**

If a user-specific ASUB is to be used for RET and/or REPOS, i.e. when:

MD11610 $MN_ASUP_EDITABLE ≠ 0

then a level of protection can be defined for the user-specific routine "_N_ASUP_SPF". The level of protection can have values in the range 0 - 7.

The setting is made via the following machine data:

MD11612 $MN_ASUP_EDIT_PROTECTION_LEVEL (level of protection of the user-specific ASUB)

For further information about protection levels, refer to:

References:
Commissioning Manual; level of protection concept

**Behavior when the single-block processing is set**

Via the following machine data it can be set that the internal ASUB or the user-specific "_N_ASUP_SPF" is processed without interruption, despite a set single block processing:

MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (prevent single-block stop)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>A stop is done in each ASUB block.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>The ASUB is processed without interruption.</td>
</tr>
</tbody>
</table>
9.11.3 Programming

Determining the cause of the ASUB activation

The cause of the activation of the ASUB can be read bit-coded via the system variable $AC_ASUP.

Continuation

When using the system ASUB, the behavior for the continuation after execution of the actions is permanently specified within the ASUB:

- System ASUB 1 → continuation with RET (subprogram return)
- System ASUB 2 → continuation with REPOS (repositioning)

The description of the system variables specifies the behavior with regard to the system ASUB for each cause at "Continued for".

Note

Continued for user-specific ASUB

It is recommended for user-specific ASUBs that the appropriate continuation of the system ASUB be retained.

Cause: Change of operating mode ($AC_ASUP, bit 9 == 1)

At a change of operating mode, the continuation depends on the machine data:

MD20114 $MC_MODESWITCH_MASK (interruption of MDA through mode change)

- Bit 0 == 0: System ASUB 1 → continuation with RET
- Bit 0 == 1: System ASUB 2 → continuation with REPOS

References

A detailed description of the system variables can be found in:
List Manual, System Variables
9.12 Perform ASUB start for user alarms

9.12.1 Function

Description

An ASUB can be initiated in various situations, either by the user, the system or event-controlled. A user alarm with the "NC Start disable" alarm response prevents the ASUB start in some situations. For a pending alarm response, a user ASUB from reset acknowledges with the alarm 16906.

With the appropriate setting of a channel-specific machine data it is possible to start and perform an ASUB although a user alarm with the alarm response "NC Start disable" is active. The alarm response is overridden for the ASUB start and permits its execution.

Note

NCK alarms with the "NC Start disable" alarm response are not affected by the override. A user ASUB from reset is still not possible and will be rejected with the alarm 16906.

User alarms that can be overridden

The number range for the user alarms that can be overridden is classified as follows:

<table>
<thead>
<tr>
<th>Number range</th>
<th>Effect</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>65000 - 65499</td>
<td>Display, NC Start disable</td>
<td>Reset</td>
</tr>
<tr>
<td>65500 - 65999</td>
<td>Display, NC Start disable (not for ASUBs for set MD20194)</td>
<td>Reset</td>
</tr>
<tr>
<td>66000 - 66999</td>
<td>Display, NC Start disable, motion standstill after executing the pre-decoded blocks</td>
<td>Reset</td>
</tr>
<tr>
<td>67000 - 67999</td>
<td>Display</td>
<td>Cancel</td>
</tr>
<tr>
<td>68000 - 68999</td>
<td>Display, NC Start disable, immediate interpolator stop</td>
<td>Reset</td>
</tr>
<tr>
<td>69000 - 69999</td>
<td>Display, NC Start disable, stop at next block end</td>
<td>Reset</td>
</tr>
</tbody>
</table>

Note

Although it is possible to override the "NC Start disable" alarm response for the following alarm number ranges, the other responses of the associated alarms ensure for the stop conditions. The stop conditions cannot be overridden with the function described in this chapter:

- 66000 – 66999
- 68000 – 68999
- 69000 – 69999
9.12.2 Activation

Setting

Each ASUB channel can be set separately with the following channel-specific machine data:

MD20194 $MC_IGNORE_NONCSTART_ASUP (ASUB start permitted despite pending alarm response "NC Start disable" for certain user alarms)

A change of the MD setting acts only with the NEWCONF part program command or from the user interface per softkey.

Sequence

The normal sequence for the ASUB start has the following form:

- Set machine data appropriate with MD20194 $MC_IGNORE_NONCSTART_ASUP and activate with NEWCONF.
- Start the part program.
  A user alarm from the number ranges that can be overridden appears, e.g. alarm 65500. This can occur from a synchronized action or with a part program command.
- Despite alarm, the part program is processed to the program end M02, M30 or M17. The Reset channel state is active.
- A started user ASUB from reset is now performed.

Note

An ASUB start from a running or stopped part program is also permitted for a pending "NC Start disable" alarm response. It is irrelevant whether a user alarm or an NCK alarm with the "NC Start disable" alarm response is involved.

Note

If the part program is stopped during the processing of the user ASUB, the ASUB can no longer be continued with the NC Start key, e.g. for M0 in the ASUB or for user stop. The "NC Start disable" alarm response is rejected with the alarm 16906. The previously generated user alarm can be acknowledged only with reset.

9.12.3 Examples

9.12.3.1 User ASUB from reset - example 1

MD20194 is not set in the application
Main program

Program code
N10 G90 G0 Z10
N20 SETAL(65500)
N30 X100
N40 Z0
N50 M30

User ASUB

Program code
N110 G91 G0 X-10 Z5
N120 X20
N130 REPOSA

Sequence

The block N10 is processed. The alarm 65500 appears that contains the "Display" and "NC Start disable" alarm responses. The part program does not stop as result. The block N30 is loaded and processed. If the user ASUB is used in the block middle, it is performed despite pending "NC Start disable" alarm response. The REPOSA resumes at the program interruption and processes the part program to the program end M30. If the user ASUB is now used, it will be rejected with alarm 16906. The NC state is reset.

9.12.3.2 User ASUB from reset - example 2

The application is the normal case, MD20194 is set.
Main program

Program code
N4 $MC_IGNORE_NONCSTART_ASUP=1
N6 NEWCONF
N10 G90 G0 Z10
N20 SETAL(65500)
N30 X100
N40 Z0
N50 M30

User ASUB

Program code
N110 G91 G0 X-10 Z5
N120 X20
N130 REPOSA

Sequence

The machine data MD20194 $MC_IGNORE_NONCSTART_ASUP is set for ASUB channel 1 and activated with NEWCONF. The block N10 is processed. The alarm 65500 appears that contains the "Display" and "NC Start disable" alarm responses. The part program does not stop as result. The block N30 is loaded and processed. If the user ASUB is used in the block middle, it is performed despite pending "NC Start disable" alarm response. The REPOSA resumes at the program interruption and processes the part program to the program end M30. If the user ASUB is now used, it is repeated because of the set machine data MD20194.

9.12.3.3 User ASUB with M0

MD20194 is set in the application.
Main program

Program code
N4 $MC_IGNORE_NONCSTART_ASUP=1
N6 NEWCONF
N10 G90 G0 Z10
N20 SETAL(65500)
N30 X100
N40 Z0
N50 M30

User ASUB

Program code
N110 G91 G0 X-10 Z5
N120 X10
N122 M0
N124 X10
N130 REPOSA

Sequence

The machine data MD20194 $MC_IGNORE_NONCSTART_ASUP is set for ASUB channel 1 and activated with NEWCONF. The block N10 is processed. The alarm 65500 appears, which contains the "Display" and "NC Start disable" alarm responses. The part program does not stop as a result. The block N30 is loaded and processed. If the user ASUB is used in the block middle, it is performed despite the pending "NC Start disable" alarm response. The part program stops at block N122 because of M0. The ASUB can no longer be continued when the NC Start key is pressed. The "NC Start disable" alarm response is rejected with the alarm 16906. The previously generated 65500 alarm can be acknowledged only with reset.
9.12.3.4 User ASUB with stop

MD20194 is set in the application.

Main program

<table>
<thead>
<tr>
<th>Program code</th>
</tr>
</thead>
<tbody>
<tr>
<td>N4 $MC_IGNORE_NONCSTART_ASUP=1</td>
</tr>
<tr>
<td>N6 NEWCONF</td>
</tr>
<tr>
<td>N10 G90 G0 Z10</td>
</tr>
<tr>
<td>N20 SETAL(65500)</td>
</tr>
<tr>
<td>N30 X100</td>
</tr>
<tr>
<td>N40 Z0</td>
</tr>
<tr>
<td>N50 M30</td>
</tr>
</tbody>
</table>

User ASUB

<table>
<thead>
<tr>
<th>Program code</th>
</tr>
</thead>
<tbody>
<tr>
<td>N110 G91 G0 X-10 Z5</td>
</tr>
<tr>
<td>N120 X20</td>
</tr>
<tr>
<td>N130 REPOSA</td>
</tr>
</tbody>
</table>

Sequence

The machine data MD20194 $MC_IGNORE_NONCSTART_ASUP is set for ASUB channel 1 and activated with NEWCONF. The block N10 is processed. The alarm 65500 appears that contains the "Display" and "NC Start disable" alarm responses. The part program does not stop as result. The block N30 is loaded and processed. If the user ASUB is used in the block middle, it is performed despite pending "NC Start disable" alarm response. If the NC Stop key is pressed for N120 in the block middle, the ASUB stops. The ASUB can no longer be continued when the NC Start key is pressed. The "NC Start disable" alarm response is rejected with the alarm 16906. The previously generated 65500 alarm can be acknowledged only with reset.
9.12.3.5 User ASUB from stopped

MD20194 is or not set in the application.

Main program

Program code

```
G90 G0 Z10
SETAL(65500)
x50
M0
X100
Z0
M30
```

User ASUB

Program code

```
G91 G0 X-10 Z5
X20
REPOSA
```

Sequence

The block N10 is processed. The alarm 65500 appears that contains the "Display" and "NC Start disable" alarm responses. The part program does not stop as result. The block N30 is loaded and processed. The part program stops at block N35 because of M0. The NC state is stopped. If the user ASUB is now used, it is performed despite the stopped NC state and pending "NC Start disable" alarm response. It does not matter whether or not the machine data MD20194 $MC_IGNORE_NONCSTART_ASUP is set. After block N120, the ASUB remains stationary before REPOSA. Repositioning is possible only at the next NC Start. The NC Start, however, is rejected with the alarm 16906 because of the "NC Start disable" alarm response. The previously generated 65500 alarm can be acknowledged only with reset.
9.13 Single block

With the "single block" function, the user can execute a part program block-by-block.

Function versions

The "single block" function is available in three versions:

- **SBL1 (= IPO single block)**
  When SBL1 is active, machining stops or pauses after each machine action block (IPO block).

- **SBL2 (= decode single block)**
  When SBL2 is active, machining always stops or pauses after each part program block. If a part program block is processed in several IPO blocks, machining stops after every IPO block. Thread cutting is an exception.

- **SBL3 (= stop in cycle)**
  As with SBL2, however, there is a stop after every part program block in the cycle, too.

Selection and activation

SBL1, SBL2, or SBL3 are selected via the user interface (below the "program control" display).

This is activated via the channel-specific NC/PLC interface signal:

DB21, ... DBX0.4 (activate single block)

9.13.1 Parameterization

Preventing single-block stop

Stopping after every block when the "single block" function is active is not desirable in some situations (e.g. during an ASUB).

The stopping at the end of a block during certain processes can be suppressed using the following machine data:

MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (prevent single-block stop)
MD20106 $MC_PROG_EVENT_IGN_SINGLEBLOCK (prog events ignore the single block)
MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP (execute interrupt program completely in spite of single block)

Sequence

If an ASUB is activated during the single block, for example, execution of the ASUB is completed. The deceleration does not take place until after the end of the ASUB or the first IPO block in which single-block suppression is not activated. If the velocity is too large for the
deceleration to be performed in this block (with continuouspath mode G64 active), further block changes are allowed.

**Note**

For decode single block SBL2, machine data MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (prevent single-block stop) is only effective with "internal ASUB", "user ASUB", and "subprograms with the attribute DISPLOF". In these cases, it is already clear at the time of interpretation that the block belongs to one of the categories cited. In these cases, further blocks can be generated.

---

### Activate debug mode for SBL2

As a result of the preprocessing of part program blocks, the reference between the current block display relative to the main run status of the NCK and the variable values displayed on the HMI can be lost. The operator display then shows implausible variable values.

With the following channel-specific setting data, you can set a preprocessing stop to be executed on every block when SBL2 is active:

SD42200 $SC_SINGLEBLOCK2_STOPRE (activate debug mode for SBL2)

This suppresses preprocessing of part program blocks and maintains the relationship between the current block display and the variable values display.

**Note**

SBL2 in debug mode does not maintain an accurate contour. In other words, as a result of the preprocessing stop, a different contour may be generated from the one created without single-block mode or with SBL1.

Debug mode for SBL2 can be used to test out part programs.

---

### 9.13.2 Programming

#### Suppress single block (SBLOF, SBLON)

The "single block" function can also be suppressed at program level:

<table>
<thead>
<tr>
<th>Language command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBLOF:</td>
<td>Switch off single-block execution</td>
</tr>
<tr>
<td></td>
<td>SBLOF can be written in a PROC block or alone in the block.</td>
</tr>
<tr>
<td>SBLON:</td>
<td>Reactivate single-block execution</td>
</tr>
<tr>
<td></td>
<td>SBLON must be in a separate block.</td>
</tr>
</tbody>
</table>
**Note**

When single-block execution SBL3 is active (stop after every part program block – even in the cycle), the SBLOF command has no effect.

**Single-block suppression for the complete program**

Programs designated with SBLOF are completely executed just like a block when single-block execution is active, i.e. single-block execution is suppressed for the complete program. SBLOF is also valid in the called subprograms.

**Single-block suppression for a subprogram**

SBLOF is in the PROC line and is valid up to the end of the subprogram or until it is interrupted. At the return command, the decision is made whether to stop at the end of the subprogram:

Return jump with M17: Stop at the end of the subprogram
Return jump with RET: No stop at the end of the subprogram

**Example:**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC EXAMPLE SBLOF</td>
<td></td>
</tr>
<tr>
<td>G1 X10</td>
<td></td>
</tr>
<tr>
<td>RET</td>
<td></td>
</tr>
</tbody>
</table>

**Single-block suppression within a program**

SBLOF alone must remain in the block. Single-block stop is deactivated from this block onwards up to the next programmed SBLON or up to the end of the active subprogram level.

**Example:**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G1 X100 F1000</td>
<td></td>
</tr>
<tr>
<td>N20 SBLOF</td>
<td>; Deactivate single block.</td>
</tr>
<tr>
<td>N30 Y20</td>
<td></td>
</tr>
<tr>
<td>N40 M100</td>
<td></td>
</tr>
<tr>
<td>N50 R10=90</td>
<td></td>
</tr>
<tr>
<td>N60 SBLON</td>
<td>; Reactivate single block.</td>
</tr>
<tr>
<td>N70 M110</td>
<td></td>
</tr>
<tr>
<td>N80 ...</td>
<td></td>
</tr>
</tbody>
</table>

The area between N20 and N60 is executed as one step in single-block mode.

**Note**

When single-block execution SBL2 is active (stop after each part program block) there is no execution stop in the SBLON block if bit 12 is set to "1" in the MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (prevent single-block stop).
Single-block suppression in the system ASUB or user ASUB

In order to execute an ASUB in one step, a PROC instruction must be programmed in the ASUB with SBLOF. This also applies to the function "Editable system ASUB" (see MD11610 $MN_ASUP_EDITABLE).

Note

The preset behavior of asynchronous subprograms with active single-block execution specified in machine data MD20117 MC_IGNORE_SINGLEBLOCK_ASUP (process interrupt program fully despite single block) can be overwritten on a program-specific basis using SBLOF.

Example:

```
<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 PROC ASUB1 SBLOF DISPLOF</td>
<td></td>
</tr>
<tr>
<td>N20 IF $AC_ASUP=='H200'</td>
<td></td>
</tr>
<tr>
<td>N30   RET</td>
<td>No REPOS for mode change.</td>
</tr>
<tr>
<td>N40 ELSE</td>
<td></td>
</tr>
<tr>
<td>N50 REPOSA</td>
<td>REPOS in all other cases.</td>
</tr>
<tr>
<td>N60 ENDIF</td>
<td></td>
</tr>
</tbody>
</table>
```

If the single-block stop in the system or user ASUB is suppressed using the settings in machine data MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (bit 0 = 1 or bit 1 = 1), then the single-block stop can be re-activated by programming SBLON in the ASUB.

If the single-block stop in the user ASUB is suppressed using the setting in machine data MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP then the single-block stop cannot be reactivated by programming SBLON in the ASUB.

Single-block suppression and block display

The current block display can be suppressed in cycles/subprograms using DISPLOF. If DISPLOF is programmed together with SBLOF, then the cycle/subprogram call continues to be displayed on single-block stops within the cycle/subprogram.

Example 1: A cycle is to act like a command for a user

Main program:

```
<table>
<thead>
<tr>
<th>Program code</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G1 X10 G90 F200</td>
</tr>
<tr>
<td>N20 X-4 Y6</td>
</tr>
<tr>
<td>N30 CYCLE1</td>
</tr>
<tr>
<td>N40 G1 X0</td>
</tr>
<tr>
<td>N50 M30</td>
</tr>
</tbody>
</table>
```

Cycle CYCLE1:

```
<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 PROC CYCLE1 DISPLOF SBLOF</td>
<td>Suppress single block</td>
</tr>
</tbody>
</table>
```
CYCLE1 is processed for an active single block, i.e. the Start key must be pressed once to process CYCLE1.

**Example 2:** An ASUB, which is started by the PLC in order to activate a modified zero offset and tool offsets, is to be executed invisibly

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 PROC ZO SBLOF DISPLOF</td>
<td></td>
</tr>
<tr>
<td>N110 CASE $P_UIFRNUM OF</td>
<td>0 GOTOF _G500</td>
</tr>
<tr>
<td></td>
<td>1 GOTOF _G54</td>
</tr>
<tr>
<td></td>
<td>2 GOTOF _G55</td>
</tr>
<tr>
<td></td>
<td>3 GOTOF _G56</td>
</tr>
<tr>
<td></td>
<td>4 GOTOF _G57</td>
</tr>
<tr>
<td></td>
<td>DEFAULT GOTOF END</td>
</tr>
<tr>
<td>N120 _G54: G54 D=$P_TOOL T=$P_TOOLNO</td>
<td></td>
</tr>
<tr>
<td>N130 RET</td>
<td></td>
</tr>
<tr>
<td>N140 _G54: G55 D=$P_TOOL T=$P_TOOLNO</td>
<td></td>
</tr>
<tr>
<td>N150 RET</td>
<td></td>
</tr>
<tr>
<td>N160 _G56: G56 D=$P_TOOL T=$P_TOOLNO</td>
<td></td>
</tr>
<tr>
<td>N170 RET</td>
<td></td>
</tr>
<tr>
<td>N180 _G57: G57 D=$P_TOOL T=$P_TOOLNO</td>
<td></td>
</tr>
<tr>
<td>N190 RET</td>
<td></td>
</tr>
<tr>
<td>N200 END: D=$P_TOOL T=$P_TOOLNO</td>
<td></td>
</tr>
<tr>
<td>N210 RET</td>
<td></td>
</tr>
</tbody>
</table>

**Single-block suppression for program nesting**

If **SBLOF** was programmed in the **PROC** instruction in a subprogram, then execution is stopped at the subprogram return jump with **M17**. That prevents the next block in the calling program from already running. If a single-block suppression is activated in a subprogram with **SBLOF**, without **SBLOF** in the **PROC** instruction, execution is only stopped after the next machine function block of the calling program. If that is not desired, **SBLON** must be programmed in the subprogram before the return (**M17**). Execution does not stop for a return jump to a higher-level program with **RET**.

**Example:**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 X0 F1000</td>
<td>; Execution is stopped in this block.</td>
</tr>
<tr>
<td>N20 UP1(0)</td>
<td></td>
</tr>
</tbody>
</table>
9.13 Single block

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC UP1(INT _NR) SBLOF</td>
<td>; Suppress single-block stop.</td>
</tr>
<tr>
<td>N100 X10</td>
<td></td>
</tr>
<tr>
<td>N110 UP2(0)</td>
<td></td>
</tr>
<tr>
<td>PROC UP2(INT _NR)</td>
<td></td>
</tr>
<tr>
<td>N200 X20</td>
<td></td>
</tr>
<tr>
<td>N210 SBLON</td>
<td>; Activate single-block stop.</td>
</tr>
<tr>
<td>N220 X22</td>
<td>; Execution is stopped in this block.</td>
</tr>
<tr>
<td>N230 UP3(0)</td>
<td></td>
</tr>
<tr>
<td>PROC UP3(INT _NR)</td>
<td></td>
</tr>
<tr>
<td>N300 SBLOF</td>
<td>; Suppress single-block stop.</td>
</tr>
<tr>
<td>N305 X30</td>
<td></td>
</tr>
<tr>
<td>N310 SBLON</td>
<td>; Activate single-block stop.</td>
</tr>
<tr>
<td>N320 X32</td>
<td>; Execution is stopped in this block.</td>
</tr>
<tr>
<td>N330 SBLOF</td>
<td>; Suppress single-block stop.</td>
</tr>
<tr>
<td>N340 X34</td>
<td></td>
</tr>
<tr>
<td>N350 M17</td>
<td>; SBLOF is active.</td>
</tr>
<tr>
<td>N240 X24</td>
<td>; Execution is stopped in this block. SBLON is active.</td>
</tr>
<tr>
<td>N250 M17</td>
<td>; Execution is stopped in this block. SBLON is active.</td>
</tr>
<tr>
<td>N120 X12</td>
<td></td>
</tr>
<tr>
<td>N130 M17</td>
<td>; Execution is stopped in this return jump block.</td>
</tr>
<tr>
<td></td>
<td>; SBLOF of the PROC instruction is active.</td>
</tr>
<tr>
<td>N30 X0</td>
<td>; Execution is stopped in this block.</td>
</tr>
<tr>
<td>N40 M30</td>
<td>; Execution is stopped in this block.</td>
</tr>
</tbody>
</table>

9.13.3 Supplementary conditions

Behavior with block-related synchronized actions

With decode single block SBL2 active, if there is a block-related synchronized action, the next stop is only executed after the next main run. The initial blocks between the synchronized action and the next main run are not stopped.

9.13.4 Single-block behavior in mode group with type A/B

Classifying channels

One mode group channel must be classified as a single-block control channel (KS), while the other mode group channels must be classified as dependent channels (KA) via interface signal. Type A or type B single-block behavior can be selected for KA channels.
Type A determines Stop (analogous to Stop key).
Type B determines Stop (analogous to stop at block limit).

### Channel classification

In **one** channel (KS) in a mode group, the user should select single block (NST DB21 ... DBX0.4 (activate single block)). Single-block type A or B refers to **other** channels (KA) of a mode group.

<table>
<thead>
<tr>
<th>Mode group</th>
<th>Channel 1 KS</th>
<th>Channel 2 KA</th>
<th>Channel 3 KA</th>
<th>...</th>
<th>Channel n KA</th>
</tr>
</thead>
</table>

Behavior can be selected via type A or type B

**Figure 9-11** Channel classification for single block in mode group 1

**Type A, IS DB11 DBX1.7=1** (single block type A)
- All channels are stopped
- All channels receive a start (Start key)
- Channel KS stops at the end of the block (due to single block)
- Channels KA receive a **STOP** (analogous to Stop key).
- All channels are stopped (deceleration phase of all KAs)

**Type B, IS DB11 DBX1.6=1** (single block type B)
- All channels are stopped
- All channels receive a start
- Channel KS stops at the end of the block
- Channels KA receive a **STOPATEND**
  (Comparable with IS DB21 to DBX7.2 (NC stop at the block limit).)
- All channels are stopped at a block limit (at some point in time)

### 9.14 Program control

#### Options

1. Function selection (via operator interface or PLC)
2. Activation of skip levels
3. Adapting the size of the interpolation buffer
4. Program display modes via an additional basic block display
9.14 Program control

5. Execution from external source (buffer size and number)
6. Execution from external subroutines

9.14.1 Function selection (via operator panel front or PLC)

User interface or PLC

The user can control part program execution via the operator panel front or PLC.

Selection, activation, feedback

Selection
Different functions are available under the Program control soft key. Selection affects an interface signal in the PLC. These signals are to be understood as selection signals from the user interface, and do not activate the selected function.

Activation
These signal states must be transferred to another area of the data block to activate the selected function. With program control by the PLC the signals are to be set directly.

Feedback
The activated functions are partly signaled back to the PLC from the NCK (see also Section “Z1: NC/PLC interface signals (Page 1551)”).

Table 9-5 Program control: Interface signals

<table>
<thead>
<tr>
<th>Function</th>
<th>Selection signal</th>
<th>Activation signal</th>
<th>Feedback signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKP skip block 0 to 7</td>
<td>DB21, ... DBX26.0 - 26.7</td>
<td>DB21, ... DBX2.0 - 2.7</td>
<td>- - -</td>
</tr>
<tr>
<td>SKP skip block 8 to 9</td>
<td>DB21, ... DBX27.0 - 27.1</td>
<td>DB21, ... DBX31.6 - 31.7</td>
<td>- - -</td>
</tr>
<tr>
<td>DRY dry run feedrate</td>
<td>DB21, ... DBX24.6</td>
<td>DB21, ... DBX0.6</td>
<td>DB21, ... DBX318.6</td>
</tr>
<tr>
<td>ROV Rapid traverse override</td>
<td>DB21, ... DBX25.3</td>
<td>DB21, ... DBX6.6</td>
<td>- - -</td>
</tr>
<tr>
<td>Single block:</td>
<td>Preselection of SBL1, SBL2 or SBL3 via program control display of HMI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBL1: Action single block</td>
<td>HMI operator panel</td>
<td>DB21, ... DBX0.4</td>
<td>- - -</td>
</tr>
<tr>
<td>SBL2: Decoding single block</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBL3: In cycle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M01 (Programmed stop)</td>
<td>DB21, ... DBX24.5</td>
<td>DB21, ... DBX0.5</td>
<td>DB21, ... DBX32.5</td>
</tr>
<tr>
<td>Associated M01</td>
<td>DB21, ... DBX24.4</td>
<td>DB21, ... DBX30.5</td>
<td>DB21, ... DBX318.5</td>
</tr>
<tr>
<td>DRF selection</td>
<td>DB21, ... DBX24.3</td>
<td>DB21, ... DBX0.3</td>
<td>DB21, ... DBX33.3</td>
</tr>
<tr>
<td>PRT program test</td>
<td>DB21, ... DBX25.7</td>
<td>DB21, ... DBX1.7</td>
<td>DB21, ... DBX33.7</td>
</tr>
</tbody>
</table>

References:
Operating Manual HMI Advanced "Operating Area, Machine"
9.14.2 Activation of skip levels

Function

It is possible to skip blocks which are not to be executed every time the program runs. Blocks to be skipped are indicated in the part program by the character "/" before the block number. The skip levels in the part program are specified by "/0" to "/9". Only one skip level can be specified for each part program block.

Parameter assignment

The number of skip levels is defined using machine data:

MD51029 $MM_MAX_SKP_LEVEL (max. number of skip levels in the NC program)

Programming

Blocks which are not to be executed in every program pass (e.g. program test blocks) can be skipped according to the following schematic.

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>/N005</td>
<td>; Block skipped, (DB21,... DBX2.0) 1st skip level</td>
</tr>
<tr>
<td>/0 N005</td>
<td>; Block skipped, (DB21,... DBX2.0) 1st skip level</td>
</tr>
<tr>
<td>/1 N010</td>
<td>; Block skipped, (DB21,... DBX2.1) 2nd skip level</td>
</tr>
<tr>
<td>/2 N020</td>
<td>; Block skipped, (DB21,... DBX2.2) 3rd skip level</td>
</tr>
<tr>
<td>/3 N030</td>
<td>; Block skipped, (DB21,... DBX2.3) 4th skip level</td>
</tr>
<tr>
<td>/4 N040</td>
<td>; Block skipped, (DB21,... DBX2.4) 5th skip level</td>
</tr>
<tr>
<td>/5 N050</td>
<td>; Block skipped, (DB21,... DBX2.5) 6th skip level</td>
</tr>
<tr>
<td>/6 N060</td>
<td>; Block skipped, (DB21,... DBX2.6) 7th skip level</td>
</tr>
<tr>
<td>/7 N070</td>
<td>; Block skipped, (DB21,... DBX2.7) 8th skip level</td>
</tr>
<tr>
<td>/8 N080</td>
<td>; Block skipped, (DB21,... DBX31.6) 9th skip level</td>
</tr>
<tr>
<td>/9 N090</td>
<td>; Block skipped, (DB21,... DBX31.7) 10th skip level</td>
</tr>
</tbody>
</table>

Activation

The 10 skip levels "/0" to "/9" are activated by the PLC setting the PLC → NCK interface signals.

The function is activated from the HMI via the "Program control" menu in the "Machine" operating area:

- For skip levels "/0" to "/7":
  Via the interface HMI → PLC DB21, ... DBB26 (skip block selected).
- For skip levels "/8" to "/9":
  Via the interface HMI → PLC DB21, ... DBX27.0 to DBX27.1.
9.14 Program control

References:
Operating Manual

Note
The levels to be skipped can only be changed when the control is in the STOP/RESET state.

9.14.3 Adapting the size of the interpolation buffer

MD28060

The channelspecific interpolator executes prepared blocks from the interpolation buffer during the part program run. The maximum number of blocks requiring space in the interpolation buffer at any given point in time is defined by the memory configuring MD28060 $MM_IPO_BUFFER_SIZE (number of NC blocks in the IPO buffer (DRAM)). For some applications it may be meaningful not to use the full buffer capacity in order to minimize the "interval" between preparation and interpolation.

SD42990

The number of blocks in the interpolation buffer can be restricted dynamically to a smaller value than in MD28060 $MC_MM_IPO_BUFFER_SIZE (number of NC blocks in the IPO buffer (DRAM)), minimum 2 blocks, with the setting data SD42990 $SC_MAX_BLOCKS_IN_IPOBUFFER (max. number of blocks in the IPO buffer).

Values of setting data SD42990 $SC_MAX_BLOCKS_IN_IPOBUFFER:

<table>
<thead>
<tr>
<th>Value</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 0</td>
<td>No interpolation buffer limit active. The max. possible interpolation buffer as set in MD 28060: MM_IPO_BUFFER_SIZE is activated.</td>
</tr>
<tr>
<td>or 1</td>
<td>The minimum permissible interpolation buffer with 2 blocks is activated.</td>
</tr>
<tr>
<td>&lt; &lt; MM_IPO_BUFFER_SIZE</td>
<td>The interpolation buffer is activated with no more than the maximum specified number of blocks.</td>
</tr>
<tr>
<td>&gt;= MM_IPO_BUFFER_SIZE</td>
<td>The interpolation buffer is activated with the number of blocks specified in MD 28060: MM_IPO_BUFFER_SIZE.</td>
</tr>
</tbody>
</table>
**Note**

If SD42990 $SC_MAX_BLOCKS_IN_IPOBUFFER is set in the part program, the interpolation buffer limitation takes effect immediately if the block with the SD is being preprocessed by the interpreter.

This means that the limitation of the IPO buffer may take effect a few blocks before the intended limitation (see also MD 28070 $MC_MM_NUM_BLOCKS_IN_PREP).

To avoid premature activation and to make the limitation of the IPO buffer take effect in synchronism with the block, a STOPRE (preprocessing stop) must be programmed before the SD is set in the part program.

**Validity**

SD42990 $SC_MAX_BLOCK_IN_IPOBUFFER has global, channel-specific validity and can also be modified in a part program. This modified value is maintained at program end. If this setting data is to be reset again on defined events, a so-called event-driven program must be created to do this. For example, this setting data could always be set to a predefined value on RESET.

**Application**

The IPO buffer limitation can be used whenever the number of blocks between block preparation and interpolation must be minimized, e.g., when actual positions in the part program must be read and processed for other purposes.

**Example**

```
N10 ...
N20 ...
............
N100 $SC_MAX_BLOCKS_IN_IPOBUFFER = 5 ; Limitation of IPO buffer to 5 NC blocks
N110 ...
N120 ...
............
N200 $SC_MAX_BLOCKS_IN_IPOBUFFER = -1 ; Cancellation of the IPO buffer limitation
N210 ...
............
```
9.14.4 Program display modes via an additional basic block display

Basic block display (only for ShopMill/ShopTurn)

A second so-called basic block display can be used with the existing block display to show all blocks that produce an action on the machine.

LookAhead basic block display

The actually approached end positions are shown as an absolute position. The position values refer either to the workpiece coordinate system (WCS) or the settable zero system (SZS).

The number of LookAhead display blocks stored in the display buffer depends on the number of prepared blocks in the NCK preprocessing buffer in the relevant processing state. If a preprocessing stop is processed, the number of display blocks is reduced to zero and increases again after the stop is acknowledged. In the case of REORG events (e.g. mode change, ASUB start), the display blocks stored for LookAhead are deleted and preprocessed again afterwards.

Processed values

Values processed in the basic block display coincide with the:

- Selected tools
- Feedrate and spindle speed
- Actually approached position values
  Exceptions:
  With active tool radius compensation, deviations can occur.
  For modulo axes, the programmed value is displayed in the basic block display. This value can also lie outside the modulo range.

Note

Generally the positions are represented in the WCS or the SZS.

The basic block display can be activated or deactivated with setting data SD42750 $SC_ABSBLOCK_ENABLE.

9.14.5 Basic block display for ShopMill/ShopTurn

Configure basic block display

The basic block display can be configured via the following machine data:

<table>
<thead>
<tr>
<th>NCK machine data for basic block display</th>
<th>MD28400 $MC_MM_ABSBLOCK</th>
<th>MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Significance:</td>
<td>Activate basic block display</td>
<td>Size of display buffer</td>
</tr>
<tr>
<td>Display machine data</td>
<td>Position values to be set:</td>
<td></td>
</tr>
</tbody>
</table>
These display machine data are copied to NCK machine data MD17200 $MN_GMMC_INFO_UNIT[0] to MD17200 $MN_GMMC_INFO_UNIT[3]. allowing them to be accessed from the NCK.

**Activating**

The basic block display is activated by MD 28400 $MC_MM_ABSBLOCK by means of Power On. If MD28400 $MC_MM_ABSBLOCK is set to 1, a channelspecific display buffer (FIFO) is created during power-up.

**Size of display buffer (FIFO)** = (MD28060 $MC_MM_IPO_BUFFER_SIZE + MD28070 $MC_MM_NUM_BLOCKS_IN_PREP) multiplied by 128 bytes. This corresponds to a size of 6KB in the machine data default setting.

Optimize size of display buffer:
The memory requirement can be optimized by entering a value between 128 and 512. The display blocks preprocessed in the display buffer are transferred to the HMI via a configurable upload buffer.

**Maximum size of upload buffer** is obtained by multiplying (MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[0] + MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[1] + 1) by the block length configured in MD28400 $MC_MM_ABSBLOCK.

The number of blocks before the current block is configured in MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[0] and the number of blocks after the current block is configured in MD28402 $MC_MM_ABSBLOCK_BUFFER_CONF[1].

**Constraints**

If the length of a display block configured in MD28400 $MC_MM_ABSBLOCK is exceeded, this display block is truncated accordingly. This is represented by string "..." at the end of the block.

For preprocessed cycles (MD10700 $MN_PREPROCESSING_LEVEL > 1), the display block contains only axis positions.

Additional boundary conditions for the basic block display:

- Modal synchronized action blocks with absolute values are not taken into account.
- The basic block display is deactivated during block search with or without computation.
- Polar coordinate programming is not shown in Cartesian system.
Radius / diameter values

Diameter values shown in the basic block display and position display may be needed as a radius for internal calculation. These values for measurements in radius/diameter according to G code group 29 can be manipulated using the following options:

- **G code DIAMCYCOF (expansion of channel-specific diameter programming)**
  This G code deactivates the channel-specific diameter programming during the cycle execution. In this way, computations in the cycle can always be done in the radius. The position display and the basic block display are continued according to the state of the diameter programming before DIAMCYCOF.
  In the basic block display, the last displayed value is retained.

- **G code DIACYCOFA[AX] (axis-specific diameter programming)**
  This G code deactivates the axis-specific diameter programming during the cycle execution. In this way, computations in the cycle can always be done in the radius. In the position display and in the basic block display, this continues according to the state of the diameter programming before DIACYCOFA[AX].
  In the basic block display, the last displayed value is retained.

- **MD27100 $MC_ABSBLOCK_FUNCTION_MASK**
  Bit0 = 1 Transverse axis setpoints are always shown as diameter values in the basic block display.

Behavior while the compressor is active

With active compressor and G/Code group 30 not equal to COMPOF, two display blocks are generated. The

- first contains the G/Code of the active compressor.
- The second contains the string "..." as character for missing display blocks.

Example:

```
G0 X10 Y10 Z10 ; Block to be preprocessed for the basic block display
COMPCAD ; Compressor for optimized surface quality (CAD prog.) A
... ; string as character for missing display blocks
```

To avoid bottlenecks in the NCK performance, the basic block display is deactivated automatically. As a sign that the display blocks are missing, a display block with the string "..." is generated.

All display blocks are always generated in the single block.
9.14.6 Structure for a DIN block

Structure of display block for a DIN block

Basic structure of display block for a DIN block

- Block number/label
- G function of the first G group
  (only if changed as compared to the last machine function block).
- Axis position
  (sequence corresponding to MD20070 $MC_AXCONF_MACHAX_USED (machine axis number valid in the channel)).
- Further modal G functions
  (only if changed as compared to the last machine function block).
- Other addresses as programmed.

The display block for the basic block display is directly derived from the programmed part program blocks according to the following rules:

- Macros are expanded.
- Skip identifiers and comments are omitted.
- Block number and labels are transferred from the original block, but omitted if DISPLOF is active.
- The number of decimal places is defined in display machine data MD 9004, MD 9010 and MD 9011 via the HMI.

<table>
<thead>
<tr>
<th>HMI display machine data</th>
<th>Access in NCK machine data</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD9004 $MM_DISPLAY_RESOLUTION</td>
<td>MD17200 $MN_GMMC_INFO_NO_UNIT[0]</td>
</tr>
<tr>
<td>MD9011 $MM_DISPLAY_RESOLUTION_INCH</td>
<td>MD17200 $MN_GMMC_INFO_NO_UNIT[1]</td>
</tr>
<tr>
<td>MD9010 $MM_SPIND_DISPLAY_RESOLUTION</td>
<td>MD17200 $MN_GMMC_INFO_NO_UNIT[2]</td>
</tr>
<tr>
<td>MD9424 $MM_MA_COORDINATE_SYSTEM</td>
<td>MD17200 $MN_GMMC_INFO_NO_UNIT[3]</td>
</tr>
</tbody>
</table>

- Programmed axis positions are represented as absolute positions in the coordinate system (WCS / ENS) specified in MD9424 $MM_MA_COORDINATE_SYSTEM (coordinate system for actual value display)

**Note**

The modulo correction is omitted for modulo axes, which means that positions outside the modulo range can be displayed. It also means that the basic block display differs from the position display in which values are always moduloconverted.
Examples

Comparisons between display block (original block) and basic block display:

- **Programmed positions** are displayed as absolute.
  The addresses AP/RP are displayed with their programmed values.

<table>
<thead>
<tr>
<th>Original block:</th>
<th>Display block:</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G90 X10.123</td>
<td>N10 X10.123</td>
</tr>
<tr>
<td>N20 G91 X1</td>
<td>N20 X11.123</td>
</tr>
</tbody>
</table>

- **Address assignments** (non-DIN addresses) are displayed in the form `<address> = <constant>`.

<table>
<thead>
<tr>
<th>Original block:</th>
<th>Display block:</th>
</tr>
</thead>
<tbody>
<tr>
<td>N110 R1 = -67.5 R2 = 7.5</td>
<td>N130 Z - 67.5 RND = 7.5</td>
</tr>
<tr>
<td>N130 Z = R1 RND = R2</td>
<td></td>
</tr>
</tbody>
</table>

- **Address indices** (address extensions) are displayed as constants `<address> [ <constant> ] = <constant>`.

<table>
<thead>
<tr>
<th>Original block:</th>
<th>Display block:</th>
</tr>
</thead>
<tbody>
<tr>
<td>N220 DEF AXIS AXIS_VAR = X</td>
<td>N240 FA[X] = 1000</td>
</tr>
<tr>
<td>N240 FA[AXIS_VAR] = R2</td>
<td></td>
</tr>
</tbody>
</table>

- **DIN addresses without address extension** are displayed in the form `<din_address> <constant>`.

<table>
<thead>
<tr>
<th>Original block:</th>
<th>Display block:</th>
</tr>
</thead>
<tbody>
<tr>
<td>N410 DEF REAL FEED = 1.5</td>
<td>N420 F1.5</td>
</tr>
<tr>
<td>N420 F = FEED</td>
<td></td>
</tr>
</tbody>
</table>
The following applies for H functions: Each programmed value is display irrespective of the output type to the PLC.

(MD22110 $MC_AUXFU_H_TYPE_INT (type of H auxiliary function is integer)).

- **For Tool selection by tool command**

  Display information is generated in the form T<value> or T=<string>. If an address extension has been programmed, this is displayed as well.

  If several spindles have been configured or the "Tool change via master toolholder" function (MD20124 $MC_TOOL_MANAGEMENT_TOOLHOLDER (toolholder number)) is active, the T number is always output with address extension.

  If no address extension has been programmed, the number of the master spindle or the master toolholder is used instead (T<spindle_number/tool_holder>= ).

- **For the Spindle programming via S, M3, M4, M5, M19, M40 - M45 and M70 (or MD 20094 $MC_SPIND_RIGID_TAPPING_M_NR (M function for switching over in the controlled axis operation))** the following regulation applies regarding the address extension:

  - If an address extension has been programmed, then this is also resolved.
  - If several spindles have been configured, then the address extension is also output.
  - If no address extension has been programmed, the number of the master spindle is used (S<spindle_number>= ).

- **Indirect G code programming** in form G[<group>] = <printout> is substituted by the corresponding G code.

<table>
<thead>
<tr>
<th>Original block:</th>
<th>Display block:</th>
</tr>
</thead>
<tbody>
<tr>
<td>N510 R1=2</td>
<td></td>
</tr>
<tr>
<td>N520 G[8]= R1</td>
<td>N520 G54</td>
</tr>
</tbody>
</table>

- **Modal G codes** that do not generate an executable block are collected and output with the display block of the next executable block if permitted by the syntax (DIN block). If this is not the case (e.g. predefined subprogram call TRANSMIT), a separate display block containing the modified G codes is placed in front of the next executable block.

<table>
<thead>
<tr>
<th>Original block:</th>
<th>Display block:</th>
</tr>
</thead>
<tbody>
<tr>
<td>N610 G64</td>
<td>G64</td>
</tr>
<tr>
<td>N620 TRANSMIT</td>
<td>N620 TRANSMIT</td>
</tr>
</tbody>
</table>

- A display block is always generated for part program lines in which the addresses F and FA appear (including for MD22240 $MC_AUXFU_F_SYNC_TYPE = 3 (output time of the F functions)).

<table>
<thead>
<tr>
<th>Original block:</th>
<th>Display block:</th>
</tr>
</thead>
<tbody>
<tr>
<td>N630 F1000</td>
<td>N630 F1000</td>
</tr>
<tr>
<td>N640 X100</td>
<td>N640 X100</td>
</tr>
</tbody>
</table>

- The display blocks generated for the block display are derived directly from the programmed part program blocks. If intermediate blocks (e.g. tool radius compensation G41/G42, radius/chamfer RNDM, RND, CHF, CHR) are generated in the course of contour preprocessing, these are assigned the display information from the part program block on which the motion is based.
• With the **EXECUTAB** command (processing a table of contour elements), the block generated by EXECUTAB is shown in the display block.

<table>
<thead>
<tr>
<th>Original block:</th>
<th>Display block:</th>
</tr>
</thead>
<tbody>
<tr>
<td>N710 Y157.5 G42</td>
<td>N710 Y157.5 G42</td>
</tr>
<tr>
<td>N720 Z-67.5 RND=7.5</td>
<td>N720 Z-67.5 RND=7.5</td>
</tr>
</tbody>
</table>

• With the **EXECSTRING** command, the block generated via EXECSTRING is displayed in the display block.

<table>
<thead>
<tr>
<th>Original block:</th>
<th>Display block:</th>
</tr>
</thead>
<tbody>
<tr>
<td>N810 EXECUTAB (KTAB[5])</td>
<td>N810 G01 X46.147 Z-25.38</td>
</tr>
</tbody>
</table>

---

**9.14.7 Execution from external**

**Function**

The "execution from external source" function can be used to execute programs from an external program memory that cannot be saved in the NC memory due to their size.

**Note**

Protected cycles (.CPF files) can **not** be executed from an external program memory.
External program memory

External program memory can be found on the following data carriers:

- Local drive
- Network drive
- USB drive

Note

Only the USB interfaces on the operator panel front or the TCU can be used as interface for the processing of an external program on a USB drive.

NOTICE

Tool/workpiece damage caused by the USB FlashDrive

It is recommended that a USB-FlashDrive is not used for the execution of an external subprogram. A communication interruption to the USB FlashDrive during the execution of the subprogram due to contact problems, failure, abort through trigger or unintentional unplugging, results in an immediate machining stop. The tool and/or workpiece could be damaged.

Applications

- Direct execution from external programs
  In principle, any program that is accessible via the directory structure of the interface in the "Execution from external" HMI mode can be selected and executed.

- Execution of external subprograms from the part program
  The external subprogram is called through the part program command `EXTCALL` with specification of a call path (optional) and the subprogram name (→ see "Executing external subprograms (EXTCALL) (Page 600)").

Parameter assignment

A reloading memory (FIFO buffer) must be reserved in the dynamic NC memory for executing a program in "execution from external source" mode.

Size of FIFO buffer

The size of the FIFO buffer is set in the machine data:

`MD18360 $MN_MM_EXT_PROG_BUFFER_SIZE` (FIFO buffer size for processing from external)

Note

Programs with jump commands

For external programs that contain jump commands (`GOTOF, GOTOB, CASE, FOR, LOOP, WHILE, REPEAT, IF, ELSE, ENDIF` etc.) the jump destinations must lie within the post loading memory.
Note

ShopMill/ShopTurn programs

The contour descriptions added at the file end mean the ShopMill and ShopTurn programs must be stored completely in the read-only memory.

Number of FIFO buffers

One FIFO buffer must be provided for each one of the programs that are executed simultaneously in "execution from external source" mode.

The number of FIFO buffers is set in the machine data:

MD18362 $MN_MM_EXT_PROG_NUM (number of externally executed program levels executable simultaneously)

Behavior on reset, power-on

External program calls are aborted using reset and power-on and the particular FIFO buffers are erased.

A main program selected from an external program memory is selected again automatically after a power-on if the same program memory is still available and execution of EXTCALL calls has been activated in MD9106 $MM_SERVER_EXTCALL_PROGRAMS.

9.14.8 Executing external subprograms (EXTCALL)

Function

Individual machining steps for producing complex workpieces may involve program sequences that require so much memory that they cannot be stored in the NC memory.

In such cases, the user has the option of executing the program sequences as subprograms from an external program memory in the "Execution from external source" mode with the help of the EXTCALL part program command.

Preconditions

The following preconditions are applicable to the execution from external subprograms:

- The subprograms must be accessible via the directory structure of the operator interface.
- A reloading memory (FIFO buffer) must be reserved for each subprogram in the dynamic NC memory.
Note
Subprograms with jump commands
For external subprograms that contain jump commands (GOTOF, GOTOB, CASE, FOR, LOOP, WHILE, REPEAT, IF, ELSE, ENDIF etc.) the jump destinations must lie within the post loading memory.

The size of the post loading memory is set via:

MD18360 MM_EXT_PROG_BUFFER_SIZE

ShopMill/ShopTurn programs
The contour descriptions added at the file end mean the ShopMill and ShopTurn programs must be stored completely in the read-only memory.

Parameterization
The path for the external subprogram directory can be preset using setting data:

SD42700 $SC_EXT_PROG_PATH (program path for the EXTCALL external subprogram call)

The entire path of the program to be called along with the subprogram path or name specified during programming is derived therefrom.

Programming
An external subprogram is called by means of parts program command EXTCALL.

Syntax:

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

Parameter:

- `<path>`: Absolute or relative path data (optional)
  Type: STRING

- `<program name>`: The program name is specified without prefix ".N.".
  The file extension ("MPF", "SPF") can be attached to program names using the "." or ":" character (optional).
  Type: STRING
Path specification: Short designations

The following short designations can be used to specify the path:

- **LOCAL_DRIVE**: for local drive
- **CF_CARD**: for CompactFlash Card
- **USB**: for USB front connection

**CF_CARD** and **LOCAL_DRIVE**: can be alternatively used.

**EXTCALL call with absolute path name**

If the subprogram exists at the specified path, it will be executed following the **EXTCALL** call. If it does not exist, program execution is canceled.

**EXTCALL call with relative path name / without path name**

In the event of an **EXTCALL** call with a relative path name or without a path name, the available program memories are searched as follows:

- If a path name is preset in SD42700 `$SC_EXT_PROG_PATH`, the data specified in the **EXTCALL** call (program name or with relative path name) is searched for first, starting from this path. The absolute path results from linking the following characters:
  - The path name preset in SD42700
  - The `/` character as a separator
  - The subprogram path or name programmed in **EXTCALL**

- If the called subprogram is not found at the preset path, the data specified in the **EXTCALL** call is then searched for in the user-memory directories.

- The search ends when the subprogram is found for the first time. If the search does not produce any hits, the program is canceled.

**Example**

**Execute from local drive**

Main program:

<table>
<thead>
<tr>
<th>Program code</th>
</tr>
</thead>
<tbody>
<tr>
<td>N010 PROC MAIN</td>
</tr>
<tr>
<td>N020 ...</td>
</tr>
<tr>
<td>N030 EXTCALL (&quot;ROUGHING&quot;)</td>
</tr>
<tr>
<td>N040 ...</td>
</tr>
<tr>
<td>N050 M30</td>
</tr>
</tbody>
</table>

External subprogram:

<table>
<thead>
<tr>
<th>Program code</th>
</tr>
</thead>
<tbody>
<tr>
<td>N010 PROC ROUGHING</td>
</tr>
</tbody>
</table>
The "MAIN.MPF" main program is stored in NC memory and is selected for execution. The "SCHRUPPEN.SPF" or "SCHRUPPEN.MPF" subprogram to be subsequently loaded is on the local drive in the directory "/user/sinumerik/data/prog/WKS.DIR/WST1.WPD".

The subprogram path is preset in SD42700:

SD42700 $SC_EXT_PROG_PATH = "LOCAL_DRIVE:WKS.DIR/WST1.WPD"

**Note**

Without the path being specified in the SD42700, the **EXTCALL** operation for this example would have to be programmed as follows:

```
EXTCALL("LOCAL_DRIVE:WKS.DIR/WST1.WPD/SCHRUPPEN")
```

### 9.15 EES (optional)

#### 9.15.1 Function

**Note**

To use the function, the "Execute from external storage (EES)" licensed option is required!

Using the EES (Execution from External Storage) function, users have the option of having the NCK execute programs **directly from an external program memory**.

The following options for external memories are available:

- Local drive LOCAL_DRIVE/CF_CARD: CF card of the NCU, user CF card of the PPU or hard disk of a CPU
- Local drive SYS_DRIVE: System CF card of the PPU

**Note**

For the SINUMERIK 828D, a user memory (100 MB) is released on the system CF card when the EES function is active. The EES function can therefore be used without a user CF card of the PPU.
- A network drive provided through Windows
- Statically managed USB drive

**NOTICE**

**Tool/workpiece damage caused by the USB FlashDrive**

A USB FlashDrive **cannot** be recommended when executing an external program. A communication interruption to the USB FlashDrive during the execution of the program due to contact problems, failure, abort through trigger or unintentional unplugging, results in an uncontrolled stop of the machining. The tool and/or workpiece could be damaged.

The EES function can replace the "Execution from external (Page 598)" and "Executing external subprograms (EXTCALL) (Page 600)" functions.

The EES function has the following advantages:
- Standard program handling throughout the system
- No restrictions regarding the commands that can be used
  The restrictions on "execution from external source" and "execution of external subprograms (EXTCALL)" e.g. no backward jumps, limitation of the jump distance of jump commands by the size of the reload memory, are eliminated with EES.
- Programs can be moved between different program memories (NC, GDIR, external drive) significantly easier.
- There are practically no restrictions regarding the part program size (this is only limited by the capacity of the external data storage).
- Uniform syntax for the subprogram call, independent of the storage location of the subprogram (an EXTCALL call is not required).
- Network drives can be used jointly by more than one station (PCU/NCU). Prerequisite is a uniform drive configuration for these stations. This results in a uniform view of the programs for all stations.
- Because of the uniform view of the external program memory for all stations, changes to the programs stored there consistently apply to all stations.

### 9.15.2 Commissioning

#### 9.15.2.1 Preconditions

To use the EES function, the drives used with the control system must be declared.

**References:**
- Commissioning Manual, base software and operating software
- Universal/Turning/Milling/Grinding Operating Manual
After activation of the new drive configuration, the programs can be freely distributed among the available drives.

**Note**
All the previous drives may no longer be available in the newly created drive configuration. Access to the programs on these drive is then no longer possible.

Remedy: First copy the programs stored on these drives to a drive that can still be accessed.

**Note**
Because protected cycles (.CPF files) are only executed from the NC part program memory for system reasons, they cannot be stored for execution on an external program memory.

The previous NC part program memory with the MPF.DIR, SPF.DIR and WKS.DIR directories is not absolutely necessary when using EES. A system can also be configured without using the NC part program memory.

**NOTICE**

**Executing programs that are not visible**
Even if the NC part program memory was removed from the drive configuration, it is always still available in the system itself. This especially means that when executing the program it is possible that programs that still exist there are inadvertently executed from the SPF.DIR.

Remedy: If the system is configured without NC part program memory in the drive configuration, any programs still stored there should be deleted manually.

If the NC part program memory is still to be used, then it should not be completely removed from the system, only assigned an appropriate protection level when required.

Drives can be shared by more than one station (PCU/NCU). A standard drive configuration for these stations means that there is a standard program view that is independent of the particular station.

**Note**
The CF card of an NCU/PPU cannot be shared by several stations.

**SINUMERIK 840D sl only**
For operation with an external HMI, the drives must be configured on the external HMI! The drive configuration (logdrive.ini) must be loaded into the corresponding NCU from the external HMI. A softkey is available for the transfer on the dialog for the drive configuration.

In systems in which multiple NCs work together, the drive configuration must be identical for all NCs. This is achieved by distributing the logdrive.ini across the corresponding NCUs.
9.15.2.2 Global part program memory (GDIR)

When declaring the drives, one of the drives can be designated the global part program memory (GDIR).

References:
Operating Manual; Section: "Managing programs" > "Setting up drives"

The system automatically creates the MPF.DIR, SPF.DIR and WKS.DIR directories on the drive that acts as the GDIR. These three directories form the GDIR.

The GDIR only plays a role for the EES function. Depending on the drive configuration, the GDIR replaces or extends the NC part program memory. However, it is not mandatory to set up a GDIR for the EES operation.

The directories and files of the GDIR can be addressed in the part program in the same way as in the passive file system. Therefore, a compatible relocation of an NC program with path specification is possible for the passive file system to the GDIR.

GDIR replaces the NC part program memory

If the NC part program memory is completely empty in the MPF.DIR, SPF.DIR and WKS.DIR directories, then the GDIR replaces the NC part program memory. The previous NCK search path is emulated 1:1 by the GDIR.

Selecting the main program on an external archive/data storage medium

Search sequence for the subprograms:
1. Actual directory on an external archive/data storage medium
2. SPF.DIR in the GDIR memory
3. The drive referenced using CALLPATH
4. Cycles

GDIR extends the NC part program memory

When the NC part program memory is not empty in the MPF.DIR, SPF.DIR and WKS.DIR directories, then the search sequence for the subprograms depends on where the main program is archived (active directory).

Selecting the main program in the NC part program memory (MPF.DIR or xxx.WPD in WKS.DIR)

Search sequence for the subprograms:
1. Actual directory in the NC part program memory
2. SPF.DIR in the NC part program memory
3. The drive referenced using CALLPATH
4. Cycles
Selecting the main program on an external archive/data storage medium released for EES

Search sequence for the subprograms:
1. Actual directory on an external archive/data storage medium
2. SPF.DIR in the NC part program memory
3. SPF.DIR in GDIR
4. The drive referenced using CALLPATH
5. Cycles

Note
To define the search sequence, also see MD11625 $MN_FILE_ONLY_WITH_EXTENSION and MD11626 $MN_CYCLES_ONLY_IN_CYCDIR!

An external drive can also be referenced using the CALLPATH statement.

9.15.2.3 Settings for file handling in the part program for EES

Program names unique throughout the system
If external program memories are used together at different stations in EES mode, file operations performed simultaneously on different stations (WRITE, DELETE, ...) result in access conflicts. To avoid such access conflicts, we recommend setting up a name space for file names on each station that is unique throughout the system.

Name space unique throughout the system
A name space for file names that is unique throughout the system is achieved, for example, by associating the file names with the EES-specific name of the NC parameterized in the machine data and the channel number of the channel in which the program is executed. For example, when the program is executed, the following code generates the file name (MYFILE_NC1_1.SPF), which is unique throughout the system, by appending the EES-specific name of the NC (NC1) and the channel number (channel 1) to the program name.

```
$MN_EES_NC_NAME="NC1"
N10 DEF STRING[31] FILENAME
N20 FILENAME="MYFILE_" << $MN_EES_NC_NAME << "=" << $P_CHANNO << ".SPF"
```

Parameterization
The EES-specific name of the NC is set in the NC-specific machine data:
**MD10125 $MN_EES_NC_NAME = <NC name>**

**Note**

**Name of the NC unique throughout the system**

To avoid access conflicts, the EES-specific name of the NC must be unique throughout the system. The responsibility for this resides exclusively with the user/machine manufacturer.

---

**When calling the program, only search for files with file ID**

In order to speed up the program search for subprogram calls during EES operation, we recommend that you limit the search to files with file ID (e.g. SPF, MPF, etc.).

**MD11625 $MN_FILE_ONLY_WITH_EXTENSION = 1**

**Note**

MD11625 has no effect on the program search when processing external subprograms with EXTCALL.

**Reference:**

Description of the search path for the subprogram call, see the Programming Manual Job Planning.

---

**Only search for programs with interface in the cycle directories**

In order to speed up the program search for subprogram calls during EES operation, we recommend the search for subprograms which have created an interface description (by means of PROC statement), and whose interface description was generated from the one of the cycle directories (CUS, CMA, CST) be limited to the cycle directories:

**MD11626 $MN_CYCLES_ONLY_IN_CYCDIR = 1**

**Note**

MD11626 has no effect on subprograms whose interface was created using an EXTERNAL declaration. A search is made in all program directories.

**NOTICE**

**No search success for cycles outside the cycle directories**

Cycles in the current directory and global subprogram directory are no longer found with the setting MD11626 = 1!

Remedy: Always store cycles in the cycle directories.
9.15.2.4 Memory configuration

Reducing the end user program memory in the passive file system

With active EES, the end user program memory in the passive file system can be reduced:

\[ \text{MD18352 $MN\_MM\_U\_FILE\_MEM\_SIZE} \] (end user memory for part programs / cycles / files)

The released memory can then be used, for example, for tool data or manufacturer cycles
\[ \text{(MD18353 $MN\_MM\_M\_FILE\_MEM\_SIZE)}. \]

References:
For detailed information on the memory configuration, see Function Manual, Extended Functions

Releasing reload memory

The EES function can replace the "Execution from external source" and "Execution of external subprograms (EXTCALL)" functions.

In order to execute subprograms from part programs with EXTCALL calls with EES instead of the "Execution of external subprograms (EXTCALL)" function, the EXTCALL calls must be changed to CALL calls, and the path specifications adapted if required.

After a complete changeover, the reload memory (FIFO buffer) required for the "Execution from external source" and "Execution of external subprograms (EXTCALL)" can be released:

\[ \text{MD18362 $MN\_MM\_EXT\_PROG\_NUM} \] (number of externally executed program levels executable simultaneously) = 0

9.16 System settings for power-up, RESET / part program end and part program start

Concept

The behavior of the control can be set via the machine data for the following events:
- Run-up (power-on)
- Reset / part program end
- Part program start

<table>
<thead>
<tr>
<th>The control-system response after:</th>
<th>Can be set with:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run-up (power on)</td>
<td>MD20110 $MC_RESET_MODE_MASK</td>
</tr>
<tr>
<td></td>
<td>MD20144 $MC_TRAFO_MODE_MASK</td>
</tr>
<tr>
<td></td>
<td>MD20150 $MC_GCODE_RESET_VALUES</td>
</tr>
<tr>
<td>RESET / part program end</td>
<td>MD20110 $MC_RESET_MODE_MASK</td>
</tr>
<tr>
<td></td>
<td>MD20150 $MC_GCODE_RESET_VALUES</td>
</tr>
<tr>
<td></td>
<td>MD20152 $MC_GCODE_RESET_MODE</td>
</tr>
<tr>
<td>The control-system response after:</td>
<td>Can be set with:</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| Part program start               | MD20112 $MC_START_MODE_MASK  
|                                  | MD20110 $MC_RESET_MODE_MASK |

* see also POWER ON (Page 783)
System settings after run-up

MD20110 $MC_RESET_MODE_MASK, bit 0 = 0 or 1

- Transformation active according to MD20140/MD20144 bit 1/MD24080 bit 0
- Geo axis exchange active according to MD20118 with MD20050

- Transformation off
- Tool length compensation off

- G codes according to MD20150 $MC_GCODE_RESET_VALUES
- Coupled-axis groupings are deactivated.
- Tangential tracking is not activated.
- Synchronous spindle coupling is not activated.
- Axial feedrate per revolution off/Path and synchronized axes on master spindle back
- Geometry axis assignment according to MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB
- Master value coupling is not activated.
- Basic frame is deselected.
- Electronic gear is not activated.
- Initial state for master spindle according to MD20090 $MC_SPIND_DEF_MASTER_SPIND
- Master tool holder initial state according to MD20124 $MC_TOOL_MANGEMENT_TOOLHOLDER
- Reference axis according to MD20100 $MC_DIAMETER_AX_DEF

RESET/parts program end

Figure 9-12 System settings after run-up (power-on)
System settings after reset / part program end and part program start

MD20110 $MC_RESET_MODE_MASK, bit 0 = 0 or 1

- **Bit 0=1**
  - G codes according to MD20152 and MD20150
  - MD20110, bits 6-18 are evaluated, i.e.
    - the function assigned to the respective bits either adopts its reset position or retains its current position (refer to tables below)

- **Bit 0=0**
  - The current settings are retained.

- **Parts program start**
  - G codes according to MD20150
  - Tool length compensation off
  - Transformation off
  - Coupled motion off
  - Tangential tracking off
  - Synchronous spindle off
  - Geometry axis assignment according to MD20050
  - Master value coupling off
  - Axial revolitional feedrate off
  - Reserved (basic frame) off
  - Master spindle according to MD20090
  - Master tool holder off
  - Reference axis according to MD20100

- **Bit ...=1**
  - Bits 4...17: Function adopts RESET position
  - Bit 18: Reference axis is retained.

- **Bit ...=0 (default)**
  - Bits 4...17: The current settings are retained.
  - Bit 18: Function adopts RESET position.

The settings activated by RESET become valid.

Interpretation of the first part program block

Figure 9-13  System settings after reset / part program end and part program start
G code effective after run-up and reset / part program end

The G code that is effective in every G group after run-up (power-on) and reset / part program end is set in the following machine data:

\[
\text{MD20150 } \$\text{MC\_GCODE\_RESET\_VALUES}\langle\text{G group}\rangle = \text{<default-G code>}
\]
\[
\text{MD20152 } \$\text{MC\_GCODE\_RESET\_MODE}\langle\text{G group}\rangle = \text{<value>}
\]

<table>
<thead>
<tr>
<th>Value</th>
<th>Description: Per G group</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The default-G code from MD20150 $MC_GCODE_RESET_VALUES takes effect.</td>
</tr>
<tr>
<td>1</td>
<td>The last active/current G code takes effect.</td>
</tr>
</tbody>
</table>

Control basic setting after run-up, reset / part program end and part program start

The control initial setting after run-up (power-on), reset / part program end and part program start is defined in the following machine data:

- MD20110 $\text{MC\_RESET\_MODE\_MASK}$ (definition of the control initial setting after run-up and reset / part program end)
- MD20112 $\text{MC\_START\_MODE\_MASK}$ (definition of the control initial setting after part program start)

References
Detailed Machine Data Description

Relevant machine data

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD20120 $\text{MC_TOOL_RESET_VALUE}$</td>
<td>Tool length compensation during run-up, reset / part program end</td>
</tr>
<tr>
<td>MD20121 $\text{MC_TOOL_PRESEL_RESET_VALUE}$</td>
<td>Preselect tool on Reset</td>
</tr>
<tr>
<td>MD20130 $\text{MC_CUTTING_EDGE_RESET_VALUE}$</td>
<td>Tool cutting-edge length compensation on run-up</td>
</tr>
<tr>
<td>MD20140 $\text{MC_TRAFO_RESET_VALUE}$</td>
<td>Run-up transformation data block</td>
</tr>
<tr>
<td>MD20144 $\text{MC_TRAFO_MODE_MASK}$</td>
<td>Selection of the kinematic transformation function</td>
</tr>
<tr>
<td>MD20150 $\text{MC_GCODE_RESET_VALUES}$</td>
<td>Initial setting of the G groups</td>
</tr>
<tr>
<td>MD20152 $\text{MC_GCODE_RESET_MODE}$</td>
<td>Reset behavior of the G groups</td>
</tr>
<tr>
<td>MD21330 $\text{MC_COUPLE_RESET_MODE_1}$</td>
<td>Coupling cancellation response</td>
</tr>
<tr>
<td>MD20050 $\text{MC_AXCONF_GEOAX_ASSIGN_TAB}$</td>
<td>Assignment of geometry axis to channel axis</td>
</tr>
<tr>
<td>MD20118 $\text{MC_GEOAX_CHANGE_RESET}$</td>
<td>Allow automatic geometry axis change</td>
</tr>
</tbody>
</table>

Example

Activate reset setting on reset:
- MD20110, bit 0 = 1
- MD20112 = 0
Transformation remains with reset / part program start:
- MD20110, bit 0 = 1
- MD20110, bit 7 = 1
- MD20112 = 0

Tool length compensation is retained after reset / part program start:
- MD20110, bit 4 = 1
- MD20110, bit 6 = 1
- MD20112 = 0

Active level (bit 4) and settable frame (bit 5) remain active after reset and are reset on part program start:
- MD20110, bit 4 = 1
- MD20110, bit 5 = 1
- MD20112, bit 4 = 1
- MD20112, bit 5 = 1

Note
MD20110/MD20112, bit 5 and bit 6
If MD20110/MD20112 are parameterized so that tool length compensation or a frame is active on a part program start in the automatic or MDI mode, the first programming of the axes must use absolute measurements (because of the traversing of the offset).

Exception: With MD42442/MD42440 the offsetting process for G91 is suppressed.

9.16.1 Tool withdrawal after POWER ON with orientation transformation

Function
If a part program with a machining operation with tool orientation is aborted due to a power failure or reset, it is possible to select the previously active transformation and generate a frame in the direction of the tool axis after the control has run up (power on). The tool can then be retracted in JOG mode by means of a retraction movement towards the tool axis.

Requirement
The active measuring systems must have a machine reference for all the machine axes involved in the transformation. See Section "Automatic restoration of the machine reference (Page 1190)".
Parameterization

The following machine data must be set so that the last active transformation is retained after POWER ON:

- MD20144 $MC_TRAFO_MODE_MASK, bit 1 = 1
- MD20110 $MC_RESET_MODE_MASK, bit 0 = 1
- MD20110 $MC_RESET_MODE_MASK, bit 7 = 1

See also Section "System settings for power-up, RESET / part program end and part program start (Page 609)".

Programming

Wait for machine reference WAITENC

With the command WAITENC, the system waits channel-specific in a program until there is a valid machine reference for all the active measuring systems of the parameterized axes. See the "Requirement" section above. The parameter assignment of the axes is performed via:

MD34800 $MA_WAIT_ENC_VALID = 1

Application

In the user program (…/_N_CMA_DIR/_N_PROG_EVENT_SPF) to be called event-controlled when running up (requirement: MD20108 bit 3 = 1), the system must wait using the command WAITENC until the valid axis positions are available.

A frame that positions the tool axis in the direction of the X, Y or Z axis can then be generated using the NC language command TOROTX/TOROTY/TOROTZ.

Example

Orientation transformation and orientation axes with incremental encoders.

<table>
<thead>
<tr>
<th>Configuration:</th>
<th>Meaning:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD10720 $MN_OPERATING_MODE_DEFAULT [0] = 6</td>
<td>Run-up in JOG mode.</td>
</tr>
<tr>
<td>MD30240 $MA_ENC_TYPE [0, &lt;axis&gt;] = 1</td>
<td>Incremental measuring system.</td>
</tr>
<tr>
<td>MD34210 $MA_ENC_REFP_STATE [0, &lt;axis&gt;] = 3</td>
<td>Enable the restoration of axis positions for incremental encoders.</td>
</tr>
<tr>
<td>MD20108 $MC_PROG_EVENT_MASK = 'H9'</td>
<td>Activate event-controlled using program PROG_EVENT during run-up and at the start of the part program.</td>
</tr>
<tr>
<td>MD20152 $MC_GCODE_RESET_MODE [52] = 1</td>
<td>Obtain TOFRAME via reset.</td>
</tr>
<tr>
<td>MD20110 $MC_RESET_MODE_MASK = 'HC1'</td>
<td>Obtain transformation and tool offset via reset.</td>
</tr>
<tr>
<td>MD20144 $MC_TRAFO_MODE_MASK = 'H02'</td>
<td>Obtain transformation via POWER OFF.</td>
</tr>
</tbody>
</table>

Event-driven user program (…/_N_CMA_DIR/_N_PROG_EVENT_SPF):

```
; Example with activation of the frame, which aligns the WCS in the tool direction, when running up and resetting with part program start.
```
The WAITENC command essentially corresponds to the following program sequence (example for 5-axis machine with AB kinematics):

```
WHILE TRUE
ENDWHILE
```

```
G4 F0.5
```

```
ENDWHILE
```

Continuing machining

**AUTOMATIC mode**

For automatic execution of programs in the AUTOMATIC mode, all the machine axes, whose actual position of the active measuring system has been restored, must be referenced.

**MDI mode and overstore**

In the MDI mode and for the overstore function, machining can also be performed, without referencing the axes, with restored positions. To do this, NC start with restored positions must be enabled explicitly for a specific channel:

```
MD20700 $MC_REFP_NC_START_LOCK = 2
```

Supplementary condition

**Axes with incremental encoders and without actual value buffering**

It is to be assumed that axes with incremental encoders and without actual value buffering are clamped with sufficient speed in the event of a power failure to prevent them drifting from their last position setpoint.
9.17 Replacing functions by subprograms

9.17.1 Overview

Function

User-specific auxiliary functions (e.g. M101) do not trigger any system functions. They are only output to the NC/PLC interface. The functionality of the auxiliary function must be implemented by the user / machine manufacturer in the PLC user program. A description will be provided as to how a user-specific subprogram call can be configured (replacement subprogram) instead of the output to NC/PLC interface, which is the default setting.

Function M101 is then still programmed in the part program. However, when executing the part program, the substitute subprogram is called. Therefore, the NC replaces the function by a subprogram call. This results in the following advantages:

- When adapting to the production process, an existing, tested and proven part program can still be used, unchanged. The changes required are then shifted into the user-specific subprograms.
- The functionality can be implemented within the substitute subprogram with the full functional scope of the NC language.
- The communication overhead between NC and PLC is not required.

Functions that can be replaced

The following functions can be replaced by subprograms:

<table>
<thead>
<tr>
<th>Auxiliary functions</th>
<th>Spindle-related functions during active synchronous spindle coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>M Switching functions</td>
<td>M40 Automatic gear stage change</td>
</tr>
<tr>
<td>T Tool selection</td>
<td>M41 - M45 Gear stage selection 1 ... 5</td>
</tr>
<tr>
<td>TCA Tool selection independent of the tool status</td>
<td>SPOS Spindle positioning</td>
</tr>
<tr>
<td>D Tool offset</td>
<td>SPOSA Spindle positioning</td>
</tr>
<tr>
<td>DL Additive tool offset</td>
<td>M19 Spindle positioning</td>
</tr>
</tbody>
</table>
9.17.2 Replacement of M, T/TCA and D/DL functions

9.17.2.1 Replacement of M functions

General Information

The following conditions are applicable for replacing the M functions:

- Per block only one M function is replaced.
- A block in which an M function is to be replaced, must not contain the following elements:
  - M98
  - Modal subprogram call
  - Subprogram return
  - Part program end
- M functions that trigger system functions must not be replaced by a subprogram (see Section “Non-replaceable M functions”).

Parameterization

M function and subprogram

M functions and the replacement subprograms are parameterized in the following machine data:

- MD10715 $MC_M_NO_FCT_CYCLE[<Index>] = <M function number>
- MD10716 $MC_M_NO_FCT_CYCLE_NAME[<Index>] = "<subprogram name>"

The M function and the corresponding replacement subprogram are connected through the same index.

Example: M function M101 is replaced by subprogram SUB_M101 and M function M102 by SUB_M102:

```
MD10715 $MC_M_NO_FCT_CYCLE[ 0 ] = 101
MD10716 $MC_M_NO_FCT_CYCLE_NAME[ 0 ] = "SUB_M101"

MD10715 $MC_M_NO_FCT_CYCLE[ 1 ] = 102
MD10716 $MC_M_NO_FCT_CYCLE_NAME[ 1 ] = "SUB_M102"
```

System variable for transferring information

For a freely selectable M function, information regarding the M function that has been replaced and additional functions (T, TCA, D, DL) for evaluation in the replacement subprogram are made available via the system variable (see Section "System variable (Page 623)"). The data contained in the system variables refers to the block in which the M function to be replaced is programmed.
The M function is selected with the index of machine data MD10715
$MC_{M\_NO\_FCT\_CYCLE}[\text{Index}]$ in which the M function to be replaced has been parameterized:

$$MD10718\ \$MC_{M\_NO\_FCT\_CYCLE\_PAR} = \text{Index}$$

### Note

For an M function replacement with transfer of information via system variable, the address extension and function value of the M function must be programmed as constant values.

**Permissible programming:**

- \(M<\text{function value}>\)
- \(M=<\text{function value}>\)
- \(M[<\text{address extension}>]=<\text{function value}>\)

**Illegal programming:**

- \(M=<\text{variable1}>\)
- \(M[<\text{variable2}>]=<\text{variable1}>\)

### Programming

Rules for replacing M functions:

- The replacement subprogram is called at the block end
- Within the replacement subprogram, no M functions are replaced
- In an ASUB, the M function is also replaced if the ASUB was started within the replacement subprogram.

### M functions that cannot be replaced

The following M functions trigger system functions as pre-defined auxiliary functions and must **not** be replaced by a subprogram:

- \(M0 \ldots M5\)
- \(M17, M30,\)
- \(M19\)
- \(M40 \ldots M45\)
- \(M98, M99\) (only for MD18800 $MN\_MM\_EXTERN\_LANGUAGE \neq 0\)

User-specific M functions parameterized via machine data must also not be replaced by a subprogram as they also trigger system functions.

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD10714 $MN_M_NO_FCT_EOP</td>
<td>M function for spindle active after RE-SET</td>
</tr>
<tr>
<td>MD10804 $MN_EXTERN_CHAN_M_NO_SET_INT</td>
<td>M function for ASUB activation (external mode)</td>
</tr>
</tbody>
</table>
### Machine data

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD10806 $MN_EXTERN_CHAN_M_NO_DISABLE_INT</td>
<td>M function for ASUB deactivation (external mode)</td>
</tr>
<tr>
<td>MD10814 $MN_EXTERN_M_NO_MAC_CYCLE</td>
<td>Macro call via M function</td>
</tr>
<tr>
<td>MD20094 $MC_SPIND_RIGID_TAPPING_M_NR</td>
<td>M function for switchover to controlled axis mode</td>
</tr>
<tr>
<td>MD20095 $MC_EXTERN_RIGID_TAPPING_M_NR</td>
<td>M function for switchover to controlled axis mode (external mode)</td>
</tr>
<tr>
<td>MD22254 $MC_AUXFU_ASSOC_M0_VALUE</td>
<td>Additional M function for program stop</td>
</tr>
<tr>
<td>MD22256 $MC_AUXFU_ASSOC_M1_VALUE</td>
<td>Additional M function for conditional stop</td>
</tr>
<tr>
<td>MD26008 $MC_NIBBLE_PUNCH_CODE</td>
<td>Definition of M functions (for nibble-specific)</td>
</tr>
<tr>
<td>MD26012 $MC_PUNCHNIB_ACTIVATION</td>
<td>Activation of punching and nibbling functions</td>
</tr>
</tbody>
</table>

### Note

**Exception**

The M function parameterized with MD22560 $MC_TOOL_CHANGE_M_CODE (tool change with M function) must not be replaced with a subprogram.

### 9.17.2.2 Replacing T/TCA and D/DL functions

#### Boundary conditions

For replacing functions T, TCA, D and DL, the following supplementary conditions apply:

- A maximum of one function replacement is active per block.

- A block with the function replacement must **not** contain the following elements:
  - M98
  - Modal subprogram call
  - Subprogram return
  - Part program end

- If the multitool slot number is programmed with address MTL for the multitool select with T = slot number, the T replacement also replaces the MTL address. The programmed values can be queried in the replacement subprogram using the $C_MTL_PROG and $C_MTL system variables.
Parameterization: Replacement subprogram

The replacement subprogram is specified function-specific in the machine data:

<table>
<thead>
<tr>
<th>Function</th>
<th>Machine data</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>MD10717 $MN_T_NO_FCT_CYCLE_NAME</td>
</tr>
<tr>
<td>TCA</td>
<td>MD15710 $MN_TCA_CYCLE_NAME</td>
</tr>
<tr>
<td>D/DL</td>
<td>MD11717 $MN_D_NO_FCT_CYCLE_NAME</td>
</tr>
</tbody>
</table>

Note

It is recommended that the same subprogram is used to replace T, TCA and D/DL functions.

Parameterization: Behavior regarding D or DL function with simultaneous T function

When D or DL and T functions are simultaneously programmed in a block, the D or DL number is either transferred as parameter to the replacement subprogram or the D or DL function is executed before calling the replacement subprogram. The behavior is configurable via:

MD10719 $MN_T_NO_FCT_CYCLE_MODE (parameterization of the T function replacement)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>The D or DL number is available in the subprogram in the form of a system variable (initial state).</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>The D or DL number is calculated directly in the block.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This function is only active if the tool change was configured with M function:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD22550 $MC_TOOL_CHANGE_MODE = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>otherwise the D or DL values are always transferred.</td>
</tr>
</tbody>
</table>

System variable for transferring information

The replacement subroutine is provided with all of the information relevant to the functions programmed in the block via system variables (see Section "System variable (Page 623)").

The data contained in the system variables refers to the block in which the function to be replaced was programmed.

Parameterization: Time that the replacement subprogram is called

The call time of the replacement subprogram is set via:
MD10719 $MN_T_NO_FCT_CYCLE_MODE, bit 1 and bit 2

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Time that the replacement subprogram is called</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>At the end of the block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>After the replacement subprogram has been executed, the interpretation is resumed with the program line following the line that triggered the replacement operation.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>At block start</td>
</tr>
<tr>
<td></td>
<td></td>
<td>After the replacement subprogram has been executed, the program line, which resulted in the replacement subprogram being called, is interpreted. The T address and the D or DL address and the M function for the tool change are no longer processed.</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>At block start and block end</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The replacement program is called twice.</td>
</tr>
</tbody>
</table>

System variable for the call time

System variable $P_SUB_STAT can be used to read whether the substitution is active, and if so, when the replacement subprogram – referred to the block – was called up:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Replacement not active</td>
</tr>
<tr>
<td>1</td>
<td>Replacement active, subprogram call is made at the block start</td>
</tr>
<tr>
<td>2</td>
<td>Replacement active, subprogram call is made at the block end</td>
</tr>
</tbody>
</table>

Example: Replacement of the T function

**Parameterization**

| MD22550 $MC_TOOL_CHANGE_MODE = 0 | Tool change with T function |
| MD10717 $MN_T_NO_FCT_CYCLE_NAME = "MY_T_CYCLE" | Name of the subprogram to replace the T function |
| MD10719 $MN_T_NO_FCT_CYCLE_MODE = 0 | Call time: End of block |

**Programming**

| N110  D1  ; D1 |
| N120  G90  G0  X100  Y100  Z50  ; D1 is active. |
| N130  D2  X110  Z0  T5  ; D1 remains active. The T function is replaced at the block end with the MY_T_CYCLE subprogram call. D2 provides MY_T_CYCLE in a system variable. |

A detailed example for replacement of the T function can be found in Section: "Examples of M/T function replacement at a tool change" (Page 624).
9.17.2.3 System variable

General Information

The replacement subprogram is provided with all of the information relevant to the functions programmed in the block (T or TCA, D or DL, M) via system variables.

Exception

D or DL number is not transferred if:

- MD10719 $MN_T_NO_FCT_CYCLE_MODE, bit 0 = 1
- MD22550 $MC_TOOL_CHANGE_MODE = 1

AND

- D or DL are programmed together with the T or M function in a block.

⚠️ CAUTION

Values do not act

The values provided for the replacement subprogram in the system variables are not yet effective. It is the sole responsibility of the user / machine manufacturer to resolve this by using the appropriate programming in the replacement subprogram.

System variable

<table>
<thead>
<tr>
<th>System variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_M_PROG</td>
<td>TRUE, if the M function has been programmed</td>
</tr>
<tr>
<td>$C_M</td>
<td>For $C_M_PROG == TRUE, contains the value of address M</td>
</tr>
</tbody>
</table>

We must differentiate between two cases here:

- $C_M supplies the value if, for the tool change with M function, a subprogram is configured with parameter transfer: MD10715 MN_M_NO_FCT_CYCLE
- If only one subprogram is configured for the addresses T and/or D/DL and if in the program the M function for the tool change is programmed together with one of the addresses to be replaced, then $C_M supplies the value: MD22560 $MC_TOOL_CHANGE_M_CODE

| $C_AUX_VALUE[0] | Value of the replaced M function |
| $C_ME           | For $C_M_PROG == TRUE, contains the value of the address extension of the M function |
| $C_AUX_EXT[0]   | Address extension of the M function (identical to $C_ME) |
| $C_AUX_IS_QUICK[0] | TRUE, if the M function was programmed with quick output to the PLC |
| $C_T_PROG       | TRUE, if the T function was programmed |
| $C_T            | For $C_T_PROG == TRUE, contains the value of the T function |
### System variable | Meaning
---|---
$C_{TE}$ | Contains for:
- $C_{T\_PROG} == TRUE$
- $C_{TS\_PROG} == TRUE$
  the value of the address extension of the T function
$C_{TS\_PROG}$ | TRUE, if for the T or TCA replacement, a tool name has been programmed.
$C_{TS}$ | For $C_{TS\_PROG} == TRUE$, contains the tool name programmed for the T or TCA replacement
$C_{TCA}$ | TRUE, if the TCA replacement is active
$C_{DUPLO\_PROG}$ | TRUE, if the duplo number of the TCA replacement has been programmed
$C_{DUPLO}$ | For $C_{DUPLO\_PROG} == TRUE$, contains the value of the programmed duplo number
$C_{THNO\_PROG}$ | TRUE, if the toolholder/spindle number of the TCA replacement has been programmed
$C_{THNO}$ | For $C_{THNO\_PROG} == TRUE$, contains the value of the programmed toolholder/spindle number
$C_{D\_PROG}$ | TRUE, if the D function has been programmed
$C_{D}$ | For $C_{D\_PROG} == TRUE$, contains the value of the D function
$C_{DL\_PROG}$ | TRUE, if the DL function was programmed
$C_{DL}$ | For $C_{DL\_PROG} == TRUE$, contains the value of the DL function
$P_{SUB\_STAT}$ | Block-related time when the replacement subprogram is called
$C_{MTL\_PROG}$ | TRUE if address MTL has been programmed
$C_{MTL}$ | For $C_{MTL\_PROG} == TRUE$, contains the value of address MTL

---

### 9.17.2.4 Example: Replacement of an M function

#### Example 1

The function M6 is replaced by calling the subprogram "SUB_M6".
The information relevant for a tool change should be transferred using system variables.

**Parameterization**

#### Machine data

| MD10715 $MN\_M\_NO\_FCT\_CYCLE[2] = 6 | Tool change with M6 |
| MD10716 $MN\_M\_NO\_FCT\_CYCLE\_NAME[2] = "SUB\_M6" | Replacement subprogram for M6 |
| MD10718 $MN\_M\_NO\_FCT\_CYCLE\_PAR = 2 | Information transfer using system variables |

**Main program**

<table>
<thead>
<tr>
<th>Programming</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC MAIN</td>
<td></td>
</tr>
</tbody>
</table>
Programming | Comment
--- | ---
... | ;
N10 T1 D1 M6 | ; M6 is replaced by subroutine "SUB_M6"
... | ;
N90 M30

**Subprogram "SUB_M6"**

<table>
<thead>
<tr>
<th>Programming</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC SUB_M6</td>
<td></td>
</tr>
</tbody>
</table>
N110 IF $C_T_PROG==TRUE | ; IF address T is programmed
N120 T[$C_TE]=$C_T | ; Execute T selection
N130 ENDIF | ; ENDIF
N140 M[$C_ME]=6 | ; Execute tool change.
N150 IF $C_D_PROG==TRUE | ; IF address D is programmed
N160 D=$C_D | ; Execute D selection
N170 ENDIF | ; ENDIF
N190 M17

**Example 2**

The new tool is prepared for changing with the T function. The tool change is only realized with function M6. The T function is replaced by calling the subprogram "MY_T_CYCLE". The D / DL number is transferred to the subprogram.

**Parameterization**

<table>
<thead>
<tr>
<th>Parameterization</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD22550 $MC_TOOL_CHANGE_MODE = 1</td>
<td>Tool change prepared with T function</td>
</tr>
<tr>
<td>MD10717 $MN_T_NO_FCT_CYCLE_NAME = &quot;MY_T_CYCLE&quot;</td>
<td>Replacement subprogram</td>
</tr>
<tr>
<td>MD10719 $MN_T_NO_FCT_CYCLE_MODE = 0</td>
<td>Transfer of the D/DL number</td>
</tr>
</tbody>
</table>

**Main program**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
</table>
| N210 D1 | ;
| N220 G90 G0 X100 Y100 Z50 | ; D1 is active.
| N230 D2 X110 Z0 T5 | ; D1 remains active, programmed D2 is transferred to the subprogram as variable |
| N240 M6 | ; Execute tool change |

**Example 3**

The new tool is prepared for changing with the T function. The tool change is only realized with function M6. The T function is replaced by calling the subprogram "MY_T_CYCLE". The D / DL number is not transferred to the subprogram.
Parameterization

<table>
<thead>
<tr>
<th>Parameterization</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD22550 $MC_TOOL_CHANGE_MODE = 1</td>
<td>Tool change prepared with T function</td>
</tr>
<tr>
<td>MD10717 $MN_T_NO_FCT_CYCLE_NAME = &quot;MY_T_CYCLE&quot;</td>
<td>Replacement subprogram</td>
</tr>
<tr>
<td>MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1</td>
<td>No transfer of the D/DL number</td>
</tr>
</tbody>
</table>

Main program

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N310 D1</td>
<td>D1 is active.</td>
</tr>
<tr>
<td>N320 G90 G0 X100 Y100 Z50 ; D1 is active.</td>
<td></td>
</tr>
<tr>
<td>N330 D2 X110 Z0 T5 ; D2 is active and is not transferred as variable to the replacement subprogram.</td>
<td></td>
</tr>
<tr>
<td>N340 M6     ; Execute tool change.</td>
<td></td>
</tr>
<tr>
<td>N330 D2 X110 Z0 T5 M6 ; D1 remains active, D2 and T5 are transferred to the M6 replacement subprogram as variable.</td>
<td></td>
</tr>
</tbody>
</table>

Example 4

The functions T and M6 are replaced by the subprogram "MY_T_CYCLE".
The parameters are transferred to the subprogram when replacing M6.
If M6 is programmed together with D or DL in the block, the D or the DL number is also transferred as parameter to the subprogram if no transfer of the D/DL number has been parameterized:
MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1

Parameterization

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD22550 $MC_TOOL_CHANGE_MODE = 1</td>
<td>Tool change with M function</td>
</tr>
<tr>
<td>MD22560 $MC_TOOL_CHANGE_M_CODE = 6</td>
<td>M code for tool change</td>
</tr>
<tr>
<td>MD10715 $MC_M_NO_FCT_CYCLE[3] = 6</td>
<td>M function to be replaced</td>
</tr>
<tr>
<td>MD10716 $MC_M_NO_FCT_CYCLE_NAME[3] = &quot;MY_T_CYCLE&quot;</td>
<td>Replacement subprogram for the M function</td>
</tr>
<tr>
<td>MD10717 $MN_T_NO_FCT_CYCLE_NAME = &quot;MY_T_CYCLE&quot;</td>
<td>Replacement subprogram for the T function</td>
</tr>
<tr>
<td>MD10718 $MN_M_NO_FCT_CYCLE_PAR = 3</td>
<td>Parameter transfer to the replacement subprogram for M6</td>
</tr>
<tr>
<td>MD10719 $MN_T_NO_FCT_CYCLE_MODE = 1</td>
<td>No transfer of the D/DL number</td>
</tr>
</tbody>
</table>

Main program

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N410 D1</td>
<td>D1 is active.</td>
</tr>
<tr>
<td>N420 G90 G0 X100 Y100 Z50 ; D1 is active.</td>
<td></td>
</tr>
<tr>
<td>N330 D2 X110 Z0 T5 M6 ; D1 remains active, D2 and T5 are transferred to the M6 replacement subprogram as variable.</td>
<td></td>
</tr>
</tbody>
</table>
9.17.2.5 Example: Replacement of a T and D function

The functions T and D are replaced by calling the subprogram "D_T_SUB_PROG". The following should also be true for the example:

- The tool change is realized with address T.
- The subprogram is called at the start of the block.
- The tool management is not active.
- Axis B is an indexing axis with Hirth gearing.

Parameterization

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD11717 $MN_D_NO_FCT_CYCLE_NAME = &quot;D_T_SUB_PROG&quot;</td>
<td>Replacement subprogram for D function</td>
</tr>
<tr>
<td>MD10717 $MN_T_NO_FCT_CYCLE_NAME = &quot;D_T_SUB_PROG&quot;</td>
<td>Replacement subprogram for M function</td>
</tr>
<tr>
<td>MD10719 $MN_T_NO_FCT_CYCLE_MODE = 'H2'</td>
<td>Call at block start</td>
</tr>
<tr>
<td>MD22550 $MC_TOOL_CHANGE_MODE = 0</td>
<td>Tool change with T function</td>
</tr>
</tbody>
</table>

Main program

```
PROC MAIN
...
N10 G01 F1000 X10 T1=5 D1
    T and D function replaced by calling "D_T_SUB_PROG" at start of block
...
N90 M30
```

Subprogram "D_T_SUB_PROG"

```
N1000 PROC D_T_SUB_PROG DISPLOF SBLOF
N4100 IF $C_T_PROG==TRUE ; IF address T is programmed
N4120    POS[B]=CAC($C_T) ; Approach the indexing position
N4130    T[$C_TE]=$C_T ; Select tool (T selection)
N4140 ENDF ; ENDF
N4300 IF $C_T_PROG==TRUE ; IF address D is programmed
N4320    D=$C_D ; Select offset (D selection)
```
9.17 Replacing functions by subprograms

### 9.17.2.6 Behavior in the event of a conflict

#### Conflict case

A conflict is present if several functions are programmed in one block and the functions should be replaced with different subprograms:

- Addresses D and DL replaced with subprogram:
  
  MD11717 $MN_FCT_CYCLE_NAME = "D_SUB_PROG"

- Address T replaced with subprogram:
  
  MD10717 $MN_FCT_CYCLE_NAME = "T_SUB_PROG"

- M function M6 replaced with subprogram:
  
  MD10715 $MN_M_NO_FCT_CYCLE[0] = 6
  MD10716 $MN_M_NO_FCT_CYCLE_NAME[0] = "M6_SUB_PROG"
  MD22550 $MC_TOOL_CHANGE_MODE = 1
  MD22560 $MC_TOOL_CHANGE_M_CODE = 6

#### Resolution

A conflict is resolved corresponding to the following table:

<table>
<thead>
<tr>
<th>The following are programmed in one program line:</th>
<th>Called subprogram:</th>
</tr>
</thead>
<tbody>
<tr>
<td>D and/or DL</td>
<td>T or TCA</td>
</tr>
<tr>
<td>___________</td>
<td>_______</td>
</tr>
<tr>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>–</td>
<td>x</td>
</tr>
<tr>
<td>–</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>–</td>
</tr>
<tr>
<td>x</td>
<td>–</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
9.17.3 Replacement of spindle functions

9.17.3.1 General

Function

When a coupling is active the following spindle functions can be replaced for leading spindles:

- M40: Automatic gear stage change
- M41 ... M45: Programmed gear stage change
- SPOS, SPOSA and M19: Spindle positioning

Boundary conditions

- To replace a spindle function, the following conditions must be met:
  - The programmed spindle must be the leading spindle of an active coupling.
  - Leading and following spindle are located in the same channel. This is only detected if the leading spindle is located in the channel in which the coupling was closed. If the leading spindle is changed to another channel, a gear stage change or positioning of this spindle does not call the replacement subprogram.
  - A programmed gear stage change must result in a real gear stage change. For this purpose, the programmed and active gear stage must differ.
- In a block, only one spindle function can be replaced. Multiple replacements lead to the termination of the program processing. The spindle functions, which are to be replaced, must then be distributed over several blocks.

Parameterization

**Spindle function**

The spindle functions to be replaced by the subprogram are selected in the machine data:

MD30465 $MA_AXIS_LANG_SUB_MASK

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Gear-stage change automatic (M40) and directly (M41-M45)</td>
</tr>
<tr>
<td></td>
<td>Value</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Spindle positioning with SPOS / SPOSA / M19</td>
</tr>
<tr>
<td></td>
<td>Value</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Subprogram: Name
The name of the replacement subprogram is entered in the machine data:

MD15700 $MN_LANG_SUB_NAME = "<subprogram name>"

Subprogram: Path
The path of the replacement subprogram is set in the machine data:

MD15702 $MN_LANG_SUB_PATH = <value>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Manufacturer cycle folder: /_N_CMA_DIR</td>
</tr>
<tr>
<td>1</td>
<td>User cycle folder: /_N_CUS_DIR</td>
</tr>
<tr>
<td>2</td>
<td>Siemens cycle folder: /_N_CST_DIR</td>
</tr>
</tbody>
</table>

System variable: Time that the replacement subprogram is called
The time that the replacement subprogram is called can be read using the system variable $P_SUB_STAT:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Replacement not active</td>
</tr>
<tr>
<td>1</td>
<td>Replacement active, subprogram call is made at the block start</td>
</tr>
<tr>
<td>2</td>
<td>Replacement active, subprogram call is made at the block end</td>
</tr>
</tbody>
</table>

Block processing
If the replacement subprogram is called at the block start, after processing the replacement subprogram, the block that initiated the call is processed. The replaced commands are no longer processed.

If the replacement subprogram is called at the block end, the block that initiated calling the replacement subprogram is first processed without the commands to be replaced. The replacement subprogram is then subsequently called.

9.17.3.2 Replacement of M40 - M45 (gear stage change)

Function
When a coupling is active, the commands for gear stage change (M40, M41 ... M45) of the leading spindle are replaced by calling a user-specific subprogram.

Parameterization

Activation
- MD30465 $MA_AXIS_LANG_SUB_MASK, bit 0 = 1
Time that the subprogram is called

- **M40**
  The time of the call cannot be set. The replacement subprogram is always called at the block start.

- **M41 ... M45**
  The call time depends on the configured output behavior of the auxiliary function to the PLC (see below MD22080):
  - Output **before** or **during** motion: Subprogram call at the **start of the block**.
  - Output **after** motion: Subprogram call at the **end of the block**

  MD22080 $MC_AUXFU_PREDEF_SPEC[12 ... 16]$ (output behavior for M41 ... M45)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>Output of the auxiliary function <strong>before</strong> the motion</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>Output of the auxiliary function <strong>during</strong> the motion</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>Output of the auxiliary function <strong>after</strong> the motion</td>
</tr>
</tbody>
</table>

System variable to transfer information

The replacement subroutine is provided with all of the information relevant to the functions programmed in the block via system variables (see Chapter "System variable (Page 632)"). The data refer exclusively to the block, in which the function to be replaced has been programmed.

9.17.3.3 Replacement of SPOS, SPOSA, M19 (spindle positioning)

Function

When a coupling is active, the positioning commands (SPOS, SPOSA or M19) of a leading spindle are replaced by calling a user-specific subprogram (replacement subprogram).

Application example

When machining workpieces in parallel on a double-spindle machine, the spindles are coupled through a coupling factor not equal to 1. When changing the tool, they must be brought to the same position. The replacement subprogram opens the coupling, separately positions the spindles at the tool change position and then recloses the coupling.

Parameterization

**Activation**

- MD30465 $MA_AXIS_LANG_SUB_MASK$, bit 1 = 1
9.17 Replacing functions by subprograms

Time that the replacement subprogram is called

- **SPOS, SPOSA**
  The time of the call cannot be set. The replacement subprogram is always called at the block start.

- **M19**
  The call time depends on the configured output behavior of the auxiliary function to the PLC (see below MD22080):
  - Output **before** or **during** motion: Subprogram call at the start of the block.
  - Output **after** motion: Subprogram call at the end of the block

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>Output of the auxiliary function <strong>before</strong> the motion</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>Output of the auxiliary function <strong>during</strong> the motion</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>Output of the auxiliary function <strong>after</strong> the motion</td>
</tr>
</tbody>
</table>

System variable for transferring information

The replacement subroutine is provided with all of the information relevant to the functions programmed in the block via system variables (see Chapter "System variable (Page 632)"). The data refer exclusively to the block, in which the function to be replaced has been programmed.

9.17.3.4 System variable

<table>
<thead>
<tr>
<th>System variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_\text{SUB_AXFCT}$</td>
<td>TRUE, if M40, M41 ... M45 replacement is active</td>
</tr>
<tr>
<td>$P_\text{SUB_GEAR}$</td>
<td>Programmed or calculated gear stage. Outside the replacement subprogram: Gear stage of the master spindle</td>
</tr>
<tr>
<td>$P_\text{SUB_AUTOGEAR}$</td>
<td>TRUE, if M40 was active in the block that had initiated the replacement operation. Outside the replacement subprogram: Actual setting in the interpreter</td>
</tr>
<tr>
<td>$P_\text{SUB_LA}$</td>
<td>Contains the axis name of the leading spindle of the active coupling, which had triggered the replacement operation. <strong>Note</strong> If the variable is used outside the replacement subprogram, program processing is cancelled with an alarm.</td>
</tr>
<tr>
<td>$P_\text{SUB_CA}$</td>
<td>Contains the axis name of the following spindle of the active coupling, which had triggered the replacement operation. <strong>Note</strong> If the variable is called outside the replacement subprogram, program processing is cancelled with an alarm.</td>
</tr>
<tr>
<td>$P_\text{SUB_AXFCT}$</td>
<td>Contains the active replacement types corresponding to MD30465 $\text{SMA_AXIS_LANG_SUB_MASK}$</td>
</tr>
<tr>
<td>$P_\text{SUB_SPOS}$</td>
<td>TRUE, if the SPOS replacement is active</td>
</tr>
</tbody>
</table>
## 9.17.3.5 Example: Gear stage change

In the subprogram, all commands to change the gear stage M40, M41 ... M45 are replaced.

### Parameterization

#### Machine data

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD15700 $MN_LANG_SUB_NAME = &quot;LANG_SUB&quot;</td>
<td>Subprogram</td>
</tr>
<tr>
<td>MD15702 $MN_LANG_SUB_PATH = 0</td>
<td>Manufacturer’s folder</td>
</tr>
<tr>
<td>MD22080 $MC_AUXFU_PREDEF_SPEC[12] = 'H21'</td>
<td>M41: Output prior to motion</td>
</tr>
<tr>
<td>MD22080 $MC_AUXFU_PREDEF_SPEC[13] = 'H21'</td>
<td>M42: Output prior to motion</td>
</tr>
<tr>
<td>MD22080 $MC_AUXFU_PREDEF_SPEC[13] = 'H21'</td>
<td>M43: Output prior to motion</td>
</tr>
<tr>
<td>MD22080 $MC_AUXFU_PREDEF_SPEC[15] = 'H21'</td>
<td>M44: Output prior to motion</td>
</tr>
<tr>
<td>MD22080 $MC_AUXFU_PREDEF_SPEC[16] = 'H21'</td>
<td>M45: Output prior to motion</td>
</tr>
<tr>
<td>MD30465 $MA_AXIS_LANG_SUB_MASK[AX5] = 'H0001'</td>
<td>Replace gear change commands</td>
</tr>
</tbody>
</table>

### Main program

<table>
<thead>
<tr>
<th>Programming</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC MAIN</td>
<td></td>
</tr>
<tr>
<td>N110 COUPON(S2,S1) ; Close the synchronous spindle coupling</td>
<td></td>
</tr>
</tbody>
</table>
**Programming**  
**Comment**

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N120</td>
<td>G01 F100 X100 S5000 M3 M43</td>
<td>Subprogram call due to M43</td>
</tr>
<tr>
<td>N120</td>
<td>M40</td>
<td>Switch-on automatic gear stage change</td>
</tr>
<tr>
<td>N140</td>
<td>M3 S1000</td>
<td>Subprogram call due to S1000</td>
</tr>
<tr>
<td>N140</td>
<td></td>
<td>as a result initiated automatic</td>
</tr>
<tr>
<td>N140</td>
<td></td>
<td>Gear stage change</td>
</tr>
<tr>
<td>N9999</td>
<td>M30</td>
<td></td>
</tr>
</tbody>
</table>

**Replacement subprogram "LANG_SUB", version 1**

Optimized for simplicity and velocity by directly addressing the spindles (S1: Leading spindle, S2: Following spindle).

<table>
<thead>
<tr>
<th>Programming</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1000 PROC LANG_SUB DISPLOF SBLOF</td>
<td></td>
</tr>
<tr>
<td>N1100 IF($P_SUB_AXFCT ==1)</td>
<td>Replacement due to gear stage change</td>
</tr>
<tr>
<td>N1140 DELAYFSTON</td>
<td>Start of stop delay area</td>
</tr>
<tr>
<td>N1150 COUPOF(S2,S1)</td>
<td>Open synchronous spindle coupling</td>
</tr>
<tr>
<td>N1160 ;gear stage change separately for leading and following spindles</td>
<td></td>
</tr>
<tr>
<td>N1170 M1=$P_SUB_GEAR M2=$P_SUB_GEAR</td>
<td></td>
</tr>
<tr>
<td>N1170 DELAYFSTON</td>
<td>End of stop delay area</td>
</tr>
<tr>
<td>N1190 COUPON(S2,S1)</td>
<td>Close the synchronous spindle coupling</td>
</tr>
<tr>
<td>N1200 ENDIF</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N9999 RET</td>
<td></td>
</tr>
</tbody>
</table>

**Replacement subprogram "LANG_SUB", version 2**

Flexibility through indirect addressing using the system variable (leading spindle: $P_SUB_LA, following spindle: $P_SUB_CA).

<table>
<thead>
<tr>
<th>Programming</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1000 PROC LANG_SUB DISPLOF SBLOF</td>
<td></td>
</tr>
<tr>
<td>N1010 DEF AXIS _LA</td>
<td>Bit memory for leading axis / leading spindle</td>
</tr>
<tr>
<td>N1020 DEF AXIS _CA</td>
<td>Bit memory for following axis / following spindle</td>
</tr>
<tr>
<td>N1030 DEF INT _GEAR</td>
<td>Bit memory for gear stage</td>
</tr>
<tr>
<td>N1100 IF($P_SUB_AXFCT==1)</td>
<td>Replacement due to gear stage change</td>
</tr>
<tr>
<td>N1110 _GEAR=$P_SUB_GEAR</td>
<td>Gear stage to be activated</td>
</tr>
<tr>
<td>N1120 _LA=$P_SUB_LA</td>
<td>Axis name of the leading spindle</td>
</tr>
<tr>
<td>N1130 _CA=$P_SUB_CA</td>
<td>Axis name of the following spindle</td>
</tr>
<tr>
<td>N1140 DELAYFSTON</td>
<td>Start of stop delay area</td>
</tr>
<tr>
<td>N1150 COUPOF(_CA,_LA)</td>
<td>Open synchronous spindle coupling</td>
</tr>
<tr>
<td>N1160 ;gear stage change for leading and following spindles</td>
<td></td>
</tr>
<tr>
<td>N1170 M[AXTOSPI(_LA)]=_GEAR M[AXTOSPI(_CA)]=_GEAR</td>
<td></td>
</tr>
</tbody>
</table>
9.17.3.6 Example: Spindle positioning

In the subprogram, only the replacement of commands \texttt{SPOS} and \texttt{SPOSA} is explicitly executed. Additional replacements should be supplemented in essentially the same fashion.

**Parameterization**

**Machine data**

- $\text{MD30465} \text{ $MA_AXIS_LANG_SUB\_MASK[AX5]} = \text{'H0002'}$
- $\text{MD22080} \text{ $MC\_AUXFU\_PREDEF\_SPEC[9]} = \text{'H0021'}$

**Setting Data**

- $\text{SD43240} \text{ $SA\_M19\_SPOS[AX5]} = 260$
- $\text{SD43250} \text{ $SA\_M19\_SPOS\_MODE[AX5]} = 4$

**Main program**

**Programming**

- \text{PROC MAIN}
- ...\text{COUPON(S2,S1) ; Activate synchronous spindle coupling}
- \text{N220 SPOS[1]=100 ; Position leading spindle with SPOS}
- ...\text{N310 G01 F1000 X100 M19 ; Position leading spindle with M19}

**Replacement subprogram "LANG\_SUB", version 1**

Optimized for simplicity and velocity by directly addressing the spindles (S1: Leading spindle, S2: Following spindle).

**Programming**

- \text{N1000 PROC LANG\_SUB DISPLOF SBLOF}
- \text{N2100 IF($P\_SUB\_AXFCT==2)}
Programming | Comment
--- | ---
N2110 | ;Replacement of SPOS/SPOSA/M19 for active synchronous spindle coupling
N2185 | DELAYFSTON ; Start of stop delay area
N2190 | COUPOF(S2,S1) ; Open synchronous spindle coupling
N2200 | ; Position leading and following spindles
N2210 | IF($P_SUB_SPOS==TRUE) OR ($P_SUB_SPOSA==TRUE)
N2220 | ; SPOS and SPOSA are mapped to SPOS
N2230 | CASE $P_SUB_SPOSMODE OF
| 0 | GOTOF LABEL1_DC \n| 1 | GOTOF LABEL1_IC \n| 2 | GOTOF LABEL1_AC \n| 3 | GOTOF LABEL1_DC \n| 4 | GOTOF LABEL1_ACP \n| 5 | GOTOF LABEL1_ACN \n| DEFAULT | GOTOF LABEL_ERR
| GOTOF LABEL1_CONT
LABEL1_IC: | DELAYFSTOF
| SPOS[1]=IC($P_SUB_SPOSIT) SPOS[2]=IC($P_SUB_SPOSIT)
| DELAYFSTON
| GOTOF LABEL1_CONT
| GOTOF LABEL1_CONT
| GOTOF LABEL1_CONT
LABEL1_CONT: | N2250 ELSE ; Positioning the spindle using M19
N2270 | M1=19 M2=19 ; Leading and following spindles
N2280 | ENDIF ; End replacement SPOS, SPOSA
N2285 | DELAYFSTOF ; End of stop delay area
N2290 | COUPOF(S2,S1) ; Activate synchronous spindle coupling
N2410 ELSE
N2420 | ; from here processing further replacements...
| ... | ...
N3300 | ENDIF ; End replacements...
| ... | ...
N9999 | RET ; Normal end of program
LABEL_ERR: | SETAL(61000) ; Error has occurred
Replacement subprogram "LANG_SUB", version 2

Flexibility through indirect addressing using the system variable (leading spindle: $P_SUB_LA, following spindle: $P_SUB_CA).

<table>
<thead>
<tr>
<th>Programming</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1000 PROC LANG_SUB DISPLOF SBLOF</td>
<td></td>
</tr>
<tr>
<td>N1010 DEF AXIS _LA ; Leading axis/spindle</td>
<td></td>
</tr>
<tr>
<td>N1020 DEF AXIS _CA ; Following axis/spindle</td>
<td></td>
</tr>
<tr>
<td>N1030 DEF INT _LSPI ; Leading spindle number (programmed ; spindle)</td>
<td></td>
</tr>
<tr>
<td>N1040 DEF INT _CSPI ; Following spindle number</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N2100 IF($P_SUB_AXFCT==2)</td>
<td></td>
</tr>
<tr>
<td>N2110 ;Replacement of SPOS/SPOSA/M19 for active synchronous spindle coupling</td>
<td></td>
</tr>
<tr>
<td>N2120 _LA=$P_SUB_LA ; Axis name of the leading spindle</td>
<td></td>
</tr>
<tr>
<td>N2130 _CA=$P_SUB_CA ; Axis name of the following spindle</td>
<td></td>
</tr>
<tr>
<td>N2140 _LSPI=AXTOSPI(_LA) ; Number of the leading spindle</td>
<td></td>
</tr>
<tr>
<td>N2180 _CSPI=AXTOSPI(_LA) ; Number of the following spindle</td>
<td></td>
</tr>
<tr>
<td>N2185 DELAYFSTON ; Start of stop delay area</td>
<td></td>
</tr>
<tr>
<td>N2190 COUPOF(_CA, _LA) ; Deactivate synchronous spindle coupling</td>
<td></td>
</tr>
<tr>
<td>N2200 ; Position leading and following spindle:</td>
<td></td>
</tr>
<tr>
<td>N2210 IF($P_SUB_SPOS==TRUE) OR ($P_SUB_SPOSA==TRUE)</td>
<td></td>
</tr>
<tr>
<td>N2220 ;SPOS and SPOSA are mapped to SPOS</td>
<td></td>
</tr>
<tr>
<td>N2230 CASE $P_SUB_SPOSMODE OF</td>
<td></td>
</tr>
<tr>
<td>0 GOTOF LABEL1_DC \</td>
<td></td>
</tr>
<tr>
<td>1 GOTOF LABEL1_IC \</td>
<td></td>
</tr>
<tr>
<td>2 GOTOF LABEL1_AC \</td>
<td></td>
</tr>
<tr>
<td>3 GOTOF LABEL1_DC \</td>
<td></td>
</tr>
<tr>
<td>4 GOTOF LABEL1_ACP \</td>
<td></td>
</tr>
<tr>
<td>5 GOTOF LABEL1_ACN \</td>
<td></td>
</tr>
<tr>
<td>DEFAULT GOTOF LABEL_ERR</td>
<td></td>
</tr>
<tr>
<td>LABEL1_DC: SPOS[_LSPI]=DC($P_SUB_SPOSIT) SPOS[_CSPI]=DC($P_SUB_SPOSIT)</td>
<td></td>
</tr>
<tr>
<td>GOTOF LABEL1_CONT</td>
<td></td>
</tr>
<tr>
<td>LABEL1_IC: DELAYFSTOF</td>
<td></td>
</tr>
<tr>
<td>SPOS[_LSPI]=IC($P_SUB_SPOSIT) SPOS[_CSPI]=IC($P_SUB_SPOSIT)</td>
<td></td>
</tr>
<tr>
<td>DELAYFSTON</td>
<td></td>
</tr>
<tr>
<td>GOTOF LABEL1_CONT</td>
<td></td>
</tr>
<tr>
<td>LABEL1_AC: SPOS[_LSPI]=AC($P_SUB_SPOSIT) SPOS[_CSPI]=AC($P_SUB_SPOSIT)</td>
<td></td>
</tr>
<tr>
<td>GOTOF LABEL1_CONT</td>
<td></td>
</tr>
<tr>
<td>LABEL1_ACP: SPOS[_LSPI]=ACP($P_SUB_SPOSIT) POS[_CSPI]=ACP($P_SUB_SPOSIT)</td>
<td></td>
</tr>
<tr>
<td>GOTOF LABEL1_CONT</td>
<td></td>
</tr>
<tr>
<td>LABEL1_ACN: SPOS[_LSPI]=ACN($P_SUB_SPOSIT) POS[_CSPI]=ACN($P_SUB_SPOSIT)</td>
<td></td>
</tr>
<tr>
<td>LABEL1_CONT:</td>
<td></td>
</tr>
<tr>
<td>N2250 ELSE ; Position the spindle using M19</td>
<td></td>
</tr>
<tr>
<td>N2270 M[_LSPI]=19 M[_CSPI]=19</td>
<td></td>
</tr>
</tbody>
</table>
9.17 Replacing functions by subprograms

### Properties of the subprograms

#### General rules

- The subprogram called when making the replacement can contain the command `PROC` and the attribute `SBLOF` and `DISPLOF`.

- The replacement is also made in the ISO language mode. However, the replacement subprograms are exclusively processed in the standard language mode (Siemens). There is an implicit switchover into the standard language mode. The original language mode is reselected with the return jump from the replacement subprogram.

- System variables are exclusively used to transfer information to the replacement subprogram. Transfer parameters are not possible.

- The behavior for a single block and attribute `SBLOF` depends on the setting in: `MD10702 IGNORE_SINGLEBLOCK_MASK`, bit 14 (prevent single-block stop)
### Value | Meaning
---|---
0 | The replacement subprogram behaves like a "normal" subprogram:
  - Return jump with \texttt{M17}: Stop at the end of the subprogram
    
    **Note**
    
    The output of the \texttt{M} function at the PLC depends on:
    - MD20800 ($MC\_SPF\_END\_TO\_VDI$, bit 0 (subprogram end to PLC))
    - Bit 0 = 0: No output
    - Bit 0 = 1: \texttt{M17} is output to the PLC.
  - Return jump with \texttt{RET}: No stop at the end of the replacement subprogram

1 | In the block, in which the replacement subprogram is called, only one stop is made. Regardless of whether:
  - The subprogram was called at the block start and/or at the block end
  - Other subprograms are called in the subprogram
  - The subprogram is exited with \texttt{M17} or \texttt{RET}

The single-block stop takes place for the replacement of \texttt{M} functions at the end of the replacement subprogram.

For the replacement of \texttt{T} and \texttt{D}/\texttt{DL} functions, the time of the single-block stop depends on when the subprogram is called:
  - Call at block start: Single-block stop at the end of the block
  - Call at the block end: Single-block stop at the end of the replacement subprogram

- For replacement subprograms with the attribute \texttt{DISPLOF} in the block display, the program line is displayed as actual block, which resulted in the subprogram being called.
- In the replacement subprogram, areas or the complete replacement subprogram can be protected against interruptions, such as NC stop, read-in inhibit etc., using the \texttt{DELAYFSTON} and \texttt{DELAYFSTOP} commands.
- Replacements do not occur recursively, i.e. the function that has led to the replacement subprogram call is no longer replaced if it is programmed again in the replacement subprogram.

#### Output of auxiliary functions to PLC

When replacing auxiliary functions, calling the replacement subprogram does not initiate that the auxiliary function is output to the PLC. The auxiliary function is only output if the auxiliary function is reprogrammed in the replacement subprogram.

#### Behavior during block search

The replacement subprogram is also called in the search modes "Block search with calculation" and "Block search with calculation in the program test mode" (SERUPRO). Any special features must be implemented in the replacement subprogram using the system variable: \texttt{$P\_SEARCH, $AC\_SERUPRO}.

Regarding collecting actions for "search with calculation", replacement subprograms behave just like "normal" subprograms.
9.17.5 Restrictions

- Function replacements are not permitted in:
  - Synchronized actions
  - Technology cycles
- There must be no blockwise synchronized actions in front of a block that contains functions at the beginning to be replaced. See the paragraph below "Example for: Non-modal synchronized actions".
- Only the actions required for the respective replacements can be performed in the replacement subprogram.
- In a block, in which the replacement subprogram is called at the block end, the following should be observed:
  - No modal subprogram call should be active
  - No subprogram return jump should be programmed
  - No program end should be programmed

Note

The controller does not monitor whether the function to be replaced has been realized in the replacement subprogram.

Example of: Non-modal synchronized actions

MD30465 $MA_AXIS_LANG_SUB_MASK, bit 0 = 1 (gear stage change)

Program code

```
N1000 WHENEVER $AA_IM[X2] <= $AA_IM[X1] + 0.5 DO $AA_OVR[X1]=0
N1010 G1 X100 M43
```

If, in block N1010, the function M43 initiates that a replacement subprogram is called, machining is interrupted and an alarm is output.

9.18 Program runtime / part counter

Information on the program runtime and workpiece counter are provided to support the machine tool operator.

This information can be processed as system variables in the NC and/or PLC program. This information is also available to be displayed on the operator interface.
9.18.1 Program runtime

Function

The "Program runtime" function provides various timers to monitor technological processes, which can be read into the part program and into synchronized actions via system variables. There are two types of timers:

1. Standard timers
   Standard timers are always active

2. Timers that can be activated
   Timers that can be activated must be activated via machine data.

Standard timers

<table>
<thead>
<tr>
<th>System variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AN_SETUP_TIME</td>
<td>Time since the last control power-up with default values in minutes.</td>
</tr>
<tr>
<td></td>
<td>Is automatically reset to &quot;0&quot; in each control power-up with default values.</td>
</tr>
<tr>
<td>$AN_POWERON_TIME</td>
<td>Time since the last normal control power-up (&quot;warm restart&quot;) in minutes.</td>
</tr>
<tr>
<td></td>
<td>Is automatically reset to &quot;0&quot; in each normal control power-up.</td>
</tr>
</tbody>
</table>
**Program runtime**

The timers to measure the program runtimes are only available in AUTOMATIC mode.

<table>
<thead>
<tr>
<th>System variable (channel-specific)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{AC_ACT_PROG_NET_TIME}$</td>
<td>Actual net runtime of the current program in seconds&lt;br&gt;Net runtime means that the time, in which the program was stopped, has been deducted.&lt;br&gt;If, in the AUTOMATIC operating mode, a part program is restarted from the RESET channel state, then $\text{AC_ACT_PROG_NET_TIME}$ is automatically reset to &quot;0&quot;.&lt;br&gt;Additional properties:&lt;br&gt;• The reset button does not reset $\text{AC_ACT_PROG_NET_TIME}$ back to &quot;0&quot;, but rather only stops the timer.&lt;br&gt;• When starting an ASUB, $\text{AC_ACT_PROG_NET_TIME}$ is set to &quot;0&quot; and also counts the runtime of the ASUB. At the end of an ASUB, it behaves just the same as for the RESET button: The timer is only held, but is not set to &quot;0&quot;.&lt;br&gt;• $\text{AC_ACT_PROG_NET_TIME}$ is not reset when starting an event-controlled program (PROG_EVENT). The program runtime is only counted further if it involves a start, M30 or a search PROG_EVENT.&lt;br&gt;• The behavior of $\text{AC_ACT_PROG_NET_TIME}$ for GOTOS and override = 0% can be parameterized using MD27850 (refer to Section &quot;Parameterization&quot;)&lt;br&gt;Tip:&lt;br&gt;With $\text{AC_PROG_NET_TIME_TRIGGER}$, $\text{AC_ACT_PROG_NET_TIME}$ can be manipulated further.</td>
</tr>
<tr>
<td>$\text{AC_OLD_PROG_NET_TIME}$</td>
<td>Net runtime in seconds of the program that has just been correctly ended&lt;br&gt;&quot;Correctly ended&quot; means that the program was not interrupted with RESET, but instead ended properly with M30.&lt;br&gt;If a new program is started, $\text{AC_OLD_PROG_NET_TIME}$ remains unscanned, till M30 is reached again.&lt;br&gt;Additional properties:&lt;br&gt;• $\text{AC_OLD_PROG_NET_TIME}$ is set to &quot;0&quot; if the currently selected program is edited.&lt;br&gt;• $\text{AC_OLD_PROG_NET_TIME}$ is not changed at the end of an ASUB or an event-controlled program (PROG_EVENT).&lt;br&gt;Tip:&lt;br&gt;The implied copying process of $\text{AC_ACT_PROG_NET_TIME}$ after $\text{AC_OLD_PROG_NET_TIME}$ takes place only when $\text{AC_PROG_NET_TIME_TRIGGER}$ is not written.</td>
</tr>
</tbody>
</table>
### System variable (channel-specific)

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$AC_OLD_PROG_NET_TIME_COUNT</strong></td>
</tr>
<tr>
<td>Changes to $AC_OLD_PROG_NET_TIME.</td>
</tr>
<tr>
<td>After POWER ON, $AC_OLD_PROG_NET_TIME_COUNT is at &quot;0&quot;.</td>
</tr>
<tr>
<td>$AC_OLD_PROG_NET_TIME_COUNT is always increased if the control has</td>
</tr>
<tr>
<td>newly written to $AC_OLD_PROG_NET_TIME.</td>
</tr>
<tr>
<td>If the user terminates the running program with RESET,</td>
</tr>
<tr>
<td>$AC_OLD_PROG_NET_TIME and $AC_OLD_PROG_NET_TIME_COUNT remain unchanged.</td>
</tr>
<tr>
<td>With $AC_OLD_PROG_NET_TIME_COUNT it can thus be ascertained, whether</td>
</tr>
<tr>
<td>$AC_OLD_PROG_NET_TIME was written.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>If two programs running consecutively have the same runtime and were ended</td>
</tr>
<tr>
<td>correctly, then the user can identify this via the changed value in</td>
</tr>
<tr>
<td>$AC_OLD_PROG_NET_TIME_COUNT.</td>
</tr>
</tbody>
</table>

| **$AC_PROG_NET_TIME_TRIGGER**                                               |
| Trigger for the selective measurement of program sections.                  |
| The function is triggered by the writing of the value to the variable:      |

<table>
<thead>
<tr>
<th>Value</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not active</td>
</tr>
<tr>
<td>1</td>
<td>Terminates the measurement</td>
</tr>
<tr>
<td></td>
<td>● $AC_OLD_PROG_NET_TIME = $AC_ACT_PROG_NET_TIME</td>
</tr>
<tr>
<td></td>
<td>● $AC_ACT_PROG_NET_TIME = 0 and then continues to run</td>
</tr>
<tr>
<td>2</td>
<td>Starts the measurement</td>
</tr>
<tr>
<td></td>
<td>● $AC_ACT_PROG_NET_TIME = 0</td>
</tr>
<tr>
<td></td>
<td>● $AC_OLD_PROG_NET_TIME is not changed</td>
</tr>
<tr>
<td>3</td>
<td>Stops the measurement</td>
</tr>
<tr>
<td></td>
<td>● $AC_OLD_PROG_NET_TIME is not changed</td>
</tr>
<tr>
<td></td>
<td>● $AC_ACT_PROG_NET_TIME is not changed</td>
</tr>
<tr>
<td>4</td>
<td>Continues the stopped measurement</td>
</tr>
<tr>
<td></td>
<td>● $AC_ACT_PROG_NET_TIME continues to run</td>
</tr>
<tr>
<td></td>
<td>● $AC_OLD_PROG_NET_TIME is not changed</td>
</tr>
</tbody>
</table>

All system variables are reset to 0 as a result of POWER ON!

### Note

**Residual time for a workpiece**

If the same workpieces are produced one after the other, then from the timer values:

- Processing time for the last workpiece produced (see $AC_OLD_PROG_NET_TIME)
- Current processing time (see $AC_ACT_PROG_NET_TIME)

the remaining residual time for a workpiece can be determined.

The residual time is displayed on the user interface in addition to the current processing time.
Note

Using STOPRE

The system variables $AC_OLD_PROG_NET_TIME and $AC_OLD_PROG_NET_TIME_COUNT do not generate any implicit preprocessing stop. This is uncritical when used in the part program if the value of the system variables comes from the previous program run. However, if the trigger for the runtime measurement ($AC_PROG_NET_TIME_TRIGGER) is written very frequently and as a result $AC_OLD_PROG_NET_TIME changes very frequently, then an explicit STOPRE should be used in the part program.

The following system variables are available for the timers that are activated/deactivated through MD parameterization:

Timers that can be activated

Program runtime

The timers to measure the program runtimes are only available in AUTOMATIC mode.

<table>
<thead>
<tr>
<th>System variable (channel-specific)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_OPERATING_TIME</td>
<td>Total runtime of NC programs in Automatic mode (in s)</td>
</tr>
<tr>
<td></td>
<td>In the automatic mode, the runtimes of all programs between NC start and end of program / reset are summed up.</td>
</tr>
<tr>
<td></td>
<td>The default is not to count in NC stop and override = 0%. Continued counting can be activated at an override of 0% via MD27860.</td>
</tr>
<tr>
<td></td>
<td>The value is automatically reset to &quot;0&quot; every time the control powers up.</td>
</tr>
<tr>
<td>$AC_CYCLE_TIME</td>
<td>Runtime of the selected NC program (in seconds)</td>
</tr>
<tr>
<td></td>
<td>The runtime between NC Start and end of program / NC reset is measured in the selected NC program.</td>
</tr>
<tr>
<td></td>
<td>The default is not to count in NC stop and override = 0%. Continued counting can be activated at an override of 0% via MD27860.</td>
</tr>
<tr>
<td></td>
<td>The value is automatically reset to &quot;0&quot; every time a new NC program starts up. MD27860 can be set to ensure that this value will be deleted even if there is a jump to the beginning of the program with GOTOS or in the event of ASUBs and PROG_EVENTs starting.</td>
</tr>
<tr>
<td>$AC_CUTTING_TIME</td>
<td>Processing time in seconds</td>
</tr>
<tr>
<td></td>
<td>The runtime of the path axes (at least one is active) is measured in all NC programs between NC Start and end of program / NC reset without rapid traverse active. MD27860 can be used to set whether measuring should only be conducted when the tool is active or independent of the tool state.</td>
</tr>
<tr>
<td></td>
<td>The measurement is interrupted when a dwell time is active.</td>
</tr>
<tr>
<td></td>
<td>The value is automatically reset to &quot;0&quot; every time the control powers up with default values.</td>
</tr>
</tbody>
</table>

Activation/deactivation

The timer that can be activated is switched-in/switched-out using machine data:
MD27860 $MC_PROCESSTIMER_MODE, bit 0 - 2 = <value>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Timer for $AC_OPERATING_TIME not active.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Timer for $AC_OPERATING_TIME active.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Timer for $AC_CYCLE_TIME not active.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Timer for $AC_CYCLE_TIME active.</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>Timer for $AC_CUTTING_TIME not active.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Timer for $AC_CUTTING_TIME active.</td>
</tr>
</tbody>
</table>

**Parameterization**

**Behavior of the timer that is always active**

The behavior of the timer that is always active for GOTOS and override = 0% is set using machine data:

MD27850 $MC_PROG_NET_TIMER_MODE

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$AC_ACT_PROG_NET_TIME is not reset to &quot;0&quot; in case of a jump with GOTOS to the program start (initial setting).</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>With a jump with GOTOS to the start of the program, $AC_ACT_PROG_NET_TIME is reset to &quot;0&quot;, the value is first saved in $AC_OLD_PROG_NET_TIME and the program counter $AC_OLD_PROG_NET_TIME_COUNT is incremented.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>For override = 0%, $AC_ACT_PROG_NET_TIME is not increased. This means that the program runtime is measured without the time for which the override was set to &quot;0&quot; (basic setting).</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Also for override = 0%, $AC_ACT_PROG_NET_TIME is increased. This means that the program runtime is measured with the time for which the override was set to &quot;0&quot;.</td>
</tr>
</tbody>
</table>

**Behavior of the timer that can be activated**

The behavior of the timer that can be activated for certain functions (e.g. test run feedrate, program test) is set using machine data:

MD27860 $MC_PROCESSTIMER_MODE

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
<td>No measurement during active dry run feedrate.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Measurement during active dry run feedrate.</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>No measurement during program test.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Measurement during program test.</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>Only for bit 1 = 1 (timer for $AC_CYCLE_TIME is active)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>$AC_CYCLE_TIME is not reset to &quot;0&quot; in case of Start through ASUB and PROG_EVENTs.</td>
</tr>
</tbody>
</table>
Supplementary conditions

- **Block search**
  No program runtimes are determined through block searches.

- **REPOS**
  The duration of a REPOS process is added to the current processing time ($AC_ACT_PROG_NET_TIME$).

Examples

Example 1: Parameterization of the runtime measurement via MD27860

- Activating the runtime measurement for the active NC program and hence no measurement in case of active dry run feedrate and program test:
  MD27860 $MC_PROCESSTIMER_MODE = 'H2'

- Activating the measurement for the tool action time and measurement also with active dry run feedrate and program test:
  MD27860 $MC_PROCESSTIMER_MODE = 'H34'

- Activating the measurement for the total runtime and the processing time with an active tool, including measurement with a program test:
  MD27860 $MC_PROCESSTIMER_MODE = 'H25'

- Activating the measurement for the total runtime and the machining time (independent of the tool), including measurement with a program test:
  MD27860 $MC_PROCESSTIMER_MODE = 'Ha5'

- Activating the measurement for the processing time with an active tool, including measurements at an override $= 0\%$, but not with a trial runfeed active:
  MD27860 $MC_PROCESSTIMER_MODE = 'H22'$

Example 2: Measuring the duration of "mySubProgrammA"

Program code

```plaintext
...  
N50 DO $AC_PROG_NET_TIME_TRIGGER=2  
N60 FOR ii= 0 TO 300
```
Program code
N70 mySubProgrammA
N80 DO $AC_PROG_NET_TIME_TRIGGER=1
N95 ENDFOR
N97 mySubProgrammB
N98 M30

After the program has processed line N80, the net runtime of "mySubProgrammA" is located in $AC_OLD_PROG_NET_TIME.

The value from $AC_OLD_PROG_NET_TIME:
- Is kept beyond M30.
- Is updated each time the loop is run through.

Example 3: Measuring the duration of "mySubProgrammA" and "mySubProgrammC"

Program code
N10 DO $AC_PROG_NET_TIME_TRIGGER=2
N20 mySubProgrammA
N30 DO $AC_PROG_NET_TIME_TRIGGER=3
N40 mySubProgrammB
N50 DO $AC_PROG_NET_TIME_TRIGGER=4
N60 mySubProgrammC
N70 DO $AC_PROG_NET_TIME_TRIGGER=1
N80 mySubProgrammD
N90 M30

9.18.2 Workpiece counter

Function
Various counters with a range of values from 0 to 999,999,999 are available with the "Workpiece counter" function in the form of channel-specific system variables. Read and write access to the system variables is possible.

The following channel-specific machine data can be used to control counter activation, counter reset timing and the counting algorithm.
System variables for workpiece counting

<table>
<thead>
<tr>
<th>System variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_REQUIRED_PARTS</td>
<td>Number of workpieces to be produced (setpoint number of workpieces)</td>
</tr>
<tr>
<td></td>
<td>In this counter the number of workpieces at which the actual workpiece count ($AC_ACTUAL_PARTS) will be reset to &quot;0&quot; can be defined.</td>
</tr>
<tr>
<td></td>
<td>MD27880 can be used to activate the generation of the display alarm: &quot;Channel %1 workpiece target = %2 reached&quot; and of the channel-specific NC/PLC interface signal: DB21, DBX317.1 (workpiece target reached)</td>
</tr>
<tr>
<td>$AC_TOTAL_PARTS</td>
<td>Total number of completed workpieces (actual workpiece total)</td>
</tr>
<tr>
<td></td>
<td>This counter specifies the total number of workpieces produced since the start time. The value is only automatically reset to &quot;0&quot; when the control runs up with default values.</td>
</tr>
<tr>
<td>$AC_ACTUAL_PARTS</td>
<td>Number of completed workpieces (actual workpiece total)</td>
</tr>
<tr>
<td></td>
<td>This counter registers the total number of workpieces produced since the start time. On condition that $AC_REQUIRED_PARTS &gt; 0, the counter is automatically reset to &quot;0&quot; when the required number of workpieces ($AC_REQUIRED_PARTS) is reached.</td>
</tr>
<tr>
<td>$AC_SPECIAL_PARTS</td>
<td>Number of workpieces selected by the user</td>
</tr>
<tr>
<td></td>
<td>This counter supports user-specific workpiece counts. An alarm can be defined to be output when the setpoint number of workpieces is reached ($AC_REQUIRED_PARTS). Users must reset the counter themselves.</td>
</tr>
</tbody>
</table>

Note
All workpiece counters are set to "0" when the control runs up with default values and can be read and written independent of their activation.

Activation

The workpiece counter is activated with the machine data:

MD27880 $MC_PART_COUNTER (activation of workpiece counters)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>$AC_REQUIRED_PARTS is active</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Alarm / signal output in case of: $AC_ACTUAL_PARTS = $AC_REQUIRED_PARTS</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Alarm / signal output in case of: $AC_SPECIAL_PARTS = $AC_REQUIRED_PARTS</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>$AC_TOTAL_PARTS is active.</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>$AC_TOTAL_PARTS is incremented by the value &quot;1&quot; through M02/M30.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>$AC_TOTAL_PARTS is incremented by the value &quot;1&quot; through the M command defined with MD27882[0].</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>$AC_TOTAL_PARTS is also active for program test / block search.</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>$AC_TOTAL_PARTS is incremented by the value &quot;1&quot; upon a jump back with GOTOS.</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>$AC_ACTUAL_PARTS is active</td>
</tr>
</tbody>
</table>
Workpiece counting with user-defined M command

If the corresponding bit is set in MD27880, then the count pulse can be triggered via a user-defined M command parameterized via the following machine data instead of via the end of program M2/M30.

$AC_ACTUAL_PARTS is incremented by the value "1" through M02/M30.

$AC_SPECIAL_PARTS is incremented by the value "1" through M02/M30.

$AC_SPECIAL_PARTS is also active for program test / block search.

$AC_SPECIAL_PARTS is incremented by the value 1 through the M command defined with MD27882[2].

The respective workpiece counter is incremented by "1", when a user-defined M command is called.

Supplementary conditions

- **Mode change / NC RESET**
  The counters are not affected by a mode change or NC RESET.

- **$AC_REQUIRED_PARTS ≤ 0**
  Where $AC_REQUIRED_PARTS ≤ 0 and MD27880 $MC_PART_COUNTER, bit 0 = 1, the counting procedure and the identity comparison set with MD27880 are not conducted for all the active counters.
Examples

- **Activation of the workpiece counter $AC_REQUIRED_PARTS**:  
  MD27880 $MC_PART_COUNTER = 'H3'  
  - $AC_REQUIRED_PARTS is active  
  - Display alarm at: $AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS

- **Activation of the workpiece counter $AC_TOTAL_PARTS**:  
  MD27880 $MC_PART_COUNTER = 'H10'  
  MD27882 $MC_PART_COUNTER_MCODE[0] = 80  
  - $AC_TOTAL_PARTS is active; the counter is incremented by the value 1 with each M02.  
  - $MC_PART_COUNTER_MCODE[0] has no significance.

- **Activation of the workpiece counter $AC_ACTUAL_PARTS**:  
  MD27880 $MC_PART_COUNTER = 'H300'  
  MD27882 $MC_PART_COUNTER_MCODE[1] = 17  
  - $AC_TOTAL_PARTS is active; the counter is incremented by a value of "1" with each M17.

- **Activation of the workpiece counter $AC_SPECIAL_PARTS**:  
  MD27880 $MC_PART_COUNTER = 'H3000'  
  MD27882 $MC_PART_COUNTER_MCODE[2] = 77  
  - $AC_TOTAL_PARTS is active; the counter is incremented by a value of "1" with each M17.

- **Deactivation of the workpiece counter $AC_ACTUAL_PARTS**:  
  MD27880 $MC_PART_COUNTER = 'H200'  
  MD27882 $MC_PART_COUNTER_MCODE[1] = 50  
  - $AC_ACTUAL_PARTS is inactive

- **Activation of all counters**:  
  MD27880 $MC_PART_COUNTER = 'H3313'  
  MD27882 $MC_PART_COUNTER_MCODE[0] = 80  
  MD27882 $MC_PART_COUNTER_MCODE[1] = 17  
  MD27882 $MC_PART_COUNTER_MCODE[2] = 77  
  - $AC_REQUIRED_PARTS is active  
  - Display alarm at: $AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS  
  - $AC_TOTAL_PARTS is active; the counter is incremented by the value 1 with each M02.  
  - $MC_PART_COUNTER_MCODE[0] has no significance.  
  - $AC_ACTUAL_PARTS is active; the counter is incremented by a value of "1" with each M17.  
  - $AC_SPECIAL_PARTS is active; the counter is incremented by a value of "1" with each M77.
- **Workpiece counter $AC_ACTUAL_PARTS is not processed during the program test / block search:**
  
  MD27880 $MC_PART_COUNTER = 'H700'
  MD27882 $MC_PART_COUNTER_MCODE[1] = 75

  - $AC_ACTUAL_PARTS is active; the counter is incremented by a value of "1" with each M75, apart from during the program test and search.

- **Cancellation of the count modes in the MD27880 $MC_PART_COUNTER with bit 0 = 1:**
  
  MD27882 $MC_PART_COUNTER_MCODE[0] = 41
  MD27882 $MC_PART_COUNTER_MCODE[1] = 42
  MD27882 $MC_PART_COUNTER_MCODE[2] = 43

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N100</td>
<td>$AC_REQUIRED_PARTS=-10 ; Value &lt; 0  □ stop counting.</td>
</tr>
<tr>
<td>N200</td>
<td>M41 M43 ; Not counting.</td>
</tr>
<tr>
<td>N300</td>
<td>M42</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N500</td>
<td>$AC_REQUIRED_PARTS=52 ; Value &gt; 0; Counting in accordance with MD27880 activated.</td>
</tr>
<tr>
<td>N501</td>
<td>M43     ; Counting.</td>
</tr>
<tr>
<td>N502</td>
<td>M42 M41 ; Counting.</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

### 9.19 Data lists

#### 9.19.1 Machine data

#### 9.19.1.1 General machine data

**Displaying machine data**

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MM_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9421</td>
<td>MA_AXES_SHOW_GEO_FIRST</td>
<td>Display geo axes of channel first</td>
</tr>
<tr>
<td>9422</td>
<td>MA_PRESET_MODE</td>
<td>PRESET / basic offset in JOG.</td>
</tr>
<tr>
<td>9423</td>
<td>MA_MAX_SKP_LEVEL</td>
<td>Maximum number of skip levels</td>
</tr>
</tbody>
</table>
### NC-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10010</td>
<td>ASSIGN_CHAN_TO_MODE_GROUP</td>
<td>Channel valid in mode group</td>
</tr>
<tr>
<td>10125</td>
<td>EES_NC_NAME</td>
<td>NCU name for the generation of unique NC program names in the EES mode</td>
</tr>
<tr>
<td>10280</td>
<td>PROG_FUNCTION_MASK</td>
<td>Comparison commands &quot;&gt;&quot; and &quot;&lt;&quot;</td>
</tr>
<tr>
<td>10700</td>
<td>PREPROCESSING_LEVEL</td>
<td>Program preprocessing level</td>
</tr>
<tr>
<td>10702</td>
<td>IGNORE_SINGLEBLOCK_MASK</td>
<td>Prevent single block stop</td>
</tr>
<tr>
<td>10707</td>
<td>PROG_TEST_MASK</td>
<td>Program test modes</td>
</tr>
<tr>
<td>10708</td>
<td>SERUPRO_MASK</td>
<td>Block change modes</td>
</tr>
<tr>
<td>10710</td>
<td>PROG_SD_RESET_SAVE_TAB</td>
<td>Setting data to be updated</td>
</tr>
<tr>
<td>10711</td>
<td>NC_LANGUAGE_CONFIGURATION</td>
<td>Manner of handling the languages, whose related option or function is not activated.</td>
</tr>
<tr>
<td>10713</td>
<td>M_NO_FCT_STOPRE</td>
<td>M function with preprocessing stop</td>
</tr>
<tr>
<td>10715</td>
<td>M_NO_FCT_CYCLE</td>
<td>M function to be replaced by subprogram</td>
</tr>
<tr>
<td>10716</td>
<td>M_NO_FCT_CYCLE_NAME</td>
<td>Subroutine name for M function replacement</td>
</tr>
<tr>
<td>10717</td>
<td>T_NO_FCT_CYCLE_NAME</td>
<td>Name of the tool change cycle for T function replacement</td>
</tr>
<tr>
<td>10718</td>
<td>M_NO_FCT_CYCLE_PAR</td>
<td>M function replacement with parameters</td>
</tr>
<tr>
<td>10719</td>
<td>T_NO_FCT_CYCLE_MODE</td>
<td>Parameter assignment for T function replacement</td>
</tr>
<tr>
<td>11450</td>
<td>SEARCH_RUN_MODE</td>
<td>Block search parameter settings</td>
</tr>
<tr>
<td>11470</td>
<td>REPOS_MODE_MASK</td>
<td>Repositioning properties</td>
</tr>
<tr>
<td>11600</td>
<td>BAG_MASK</td>
<td>Mode group response to ASUB</td>
</tr>
<tr>
<td>11602</td>
<td>ASUP_START_MASK</td>
<td>Ignore stop conditions for ASUB</td>
</tr>
<tr>
<td>11604</td>
<td>ASUP_START_Prio_LEVEL</td>
<td>Priorities, starting from which $MN_ASUP_START_MASK is effective</td>
</tr>
<tr>
<td>11610</td>
<td>ASUP_EDITABLE</td>
<td>Activating a user-specific ASUB program</td>
</tr>
<tr>
<td>11612</td>
<td>ASUP_EDIT_PROTECTION_LEVEL</td>
<td>Protection level of user-specific ASUB program</td>
</tr>
<tr>
<td>11620</td>
<td>PROG_EVENT_NAME</td>
<td>Program name for PROG_EVENT</td>
</tr>
<tr>
<td>11625</td>
<td>FILE ONLY WITH EXTENSION</td>
<td>When calling the program, only search for files with a file ID</td>
</tr>
<tr>
<td>11626</td>
<td>CYCLES ONLY IN CYCDIR</td>
<td>Only search for programs with interface in the cycle directories</td>
</tr>
<tr>
<td>11717</td>
<td>D_NO_FCT_CYCLE_NAME</td>
<td>Subroutine name for D function replacement</td>
</tr>
<tr>
<td>15700</td>
<td>LANG_SUB_NAME</td>
<td>Name for replacement subprogram</td>
</tr>
<tr>
<td>15702</td>
<td>LANG_SUB_PATH</td>
<td>Call path for replacement subprogram</td>
</tr>
<tr>
<td>17200</td>
<td>GMMC_INFO_NO_UNIT</td>
<td>Global HMI info (without physical unit)</td>
</tr>
<tr>
<td>17201</td>
<td>GMMC_INFO_NO_UNIT_STATUS</td>
<td>Global HMI status info (without physical unit)</td>
</tr>
<tr>
<td>18045</td>
<td>EES_MODE_INFO</td>
<td>Mode in which the EES function operates</td>
</tr>
<tr>
<td>18360</td>
<td>MM_EXT_PROG_BUFFER_SIZE</td>
<td>FIFO buffer size for execution from external source (DRAM)</td>
</tr>
<tr>
<td>18362</td>
<td>MM_EXT_PROG_NUM</td>
<td>Number of program levels that can be processed simultaneously from external</td>
</tr>
</tbody>
</table>
9.19.1.2 Channel-specific machine data

Basic machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20000</td>
<td>CHAN_NAME</td>
<td>Channel name</td>
</tr>
<tr>
<td>20050</td>
<td>AXCONF_GEOAX_ASSIGN_TAB</td>
<td>Assignment of geometry axis to channel axis</td>
</tr>
<tr>
<td>20060</td>
<td>AXCONF_GEOAX_NAME_TAB</td>
<td>Geometry axis name in channel</td>
</tr>
<tr>
<td>20070</td>
<td>AXCONF_MACHAX_USED</td>
<td>Machine axis number valid in channel</td>
</tr>
<tr>
<td>20080</td>
<td>AXCONF_CHANAX_NAME_TAB</td>
<td>Channel axis name in channel [channel axis no.]: 0...7</td>
</tr>
<tr>
<td>20090</td>
<td>SPIND_DEF_MASTER_SPIND</td>
<td>Initial setting of master spindle in channel</td>
</tr>
<tr>
<td>20100</td>
<td>DIAMETER_AX_DEF</td>
<td>Geometry axis with transverse axis function</td>
</tr>
<tr>
<td>20106</td>
<td>PROG_EVENT_IGN_SINGLEBLOCK</td>
<td>Prog events ignore the single block</td>
</tr>
<tr>
<td>20107</td>
<td>PROG_EVENT_IGN_INHIBIT</td>
<td>Prog events ignore the read-in disable</td>
</tr>
<tr>
<td>20108</td>
<td>PROG_EVENT_MASK</td>
<td>Event-driven program calls</td>
</tr>
<tr>
<td>20109</td>
<td>PROG_EVENT_MASK_PROPERTIES</td>
<td>Prog events properties</td>
</tr>
<tr>
<td>20114</td>
<td>MODESWITCH_MASK</td>
<td>Setting for REPOS</td>
</tr>
<tr>
<td>20116</td>
<td>IGNORE_INHIBIT_ASUP</td>
<td>Execute user ASUBs completely in spite of read-in disable</td>
</tr>
<tr>
<td>20117</td>
<td>IGNORE_SINGLEBLOCK_ASUP</td>
<td>Process user ASUBs completely in spite of single-block processing</td>
</tr>
<tr>
<td>20160</td>
<td>CUBIC_SPLINE_BLOCKS</td>
<td>Number of blocks for C spline</td>
</tr>
<tr>
<td>20170</td>
<td>COMPRESS_BLOCK_PATH_LIMIT</td>
<td>Maximum traversing length of NC block for compression</td>
</tr>
<tr>
<td>20191</td>
<td>IGN_PROG_STATE_ASUP</td>
<td>Do not display the execution of the interrupt routine on the operator panel</td>
</tr>
<tr>
<td>20192</td>
<td>PROG_EVENT_IGN_PROG_STATE</td>
<td>Do not display the execution of the program events on the operator panel</td>
</tr>
<tr>
<td>20193</td>
<td>PROG_EVENT_IGN_STOP</td>
<td>Prog events ignore the Stop key</td>
</tr>
<tr>
<td>20194</td>
<td>IGNORE_NONCSTART_ASUP</td>
<td>ASUB start permitted despite pending alarm response “NC Start disable” for certain user alarms</td>
</tr>
<tr>
<td>20210</td>
<td>CUTCOM_CORNER_LIMIT</td>
<td>Max. angle for intersection calculation with tool radius compensation</td>
</tr>
<tr>
<td>20220</td>
<td>CUTCOM_MAX_DISC</td>
<td>Maximum value for DISC</td>
</tr>
<tr>
<td>20230</td>
<td>CUTCOM_CURVE_INSERT_LIMIT</td>
<td>Maximum angle for intersection calculation with tool radius compensation</td>
</tr>
<tr>
<td>20240</td>
<td>CUTCOM_MAXNUM_CHECK_BLOCKS</td>
<td>Blocks for predictive contour calculation with tool radius compensation</td>
</tr>
<tr>
<td>20250</td>
<td>CUTCOM_MAXNUM_DUMMY_BLOCKS</td>
<td>Maximum number of blocks without traversing motion for TRC</td>
</tr>
<tr>
<td>20270</td>
<td>CUTTING_EDGE_DEFAULT</td>
<td>Basic setting of tool cutting edge without programming</td>
</tr>
<tr>
<td>20400</td>
<td>LOOKAH_USE_VELO_NEXT_BLOCK</td>
<td>Look Ahead to programmed following block velocity</td>
</tr>
<tr>
<td>20430</td>
<td>LOOKAH_NUM_OVR_POINTS</td>
<td>Number of override corner values for Look Ahead</td>
</tr>
<tr>
<td>20440</td>
<td>LOOKAH_OVR_POINTS</td>
<td>Override switch points for LookAhead</td>
</tr>
<tr>
<td>20500</td>
<td>CONST VELO MIN_TIME</td>
<td>Minimum time with constant velocity</td>
</tr>
</tbody>
</table>
## Number Identifier: $MC_

### Description

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20600</td>
<td>MAX_PATH_JERK</td>
<td>Path-related maximum jerk</td>
</tr>
<tr>
<td>20610</td>
<td>ADD_MOVE_ACCEL_RESERVE</td>
<td>Acceleration reserve for overlaid motions</td>
</tr>
<tr>
<td>20700</td>
<td>REFP_NC_START_LOCK</td>
<td>NC start disable without reference point</td>
</tr>
<tr>
<td>20750</td>
<td>ALLOW_GO_IN_G96</td>
<td>G0 logic for G96, G961</td>
</tr>
<tr>
<td>20800</td>
<td>SPF_END_TO_VDI</td>
<td>Subprogram end to PLC</td>
</tr>
<tr>
<td>21000</td>
<td>CIRCLE_ERROR_CONST</td>
<td>Circle end point monitoring constant</td>
</tr>
<tr>
<td>21010</td>
<td>CIRCLE_ERROR_FACTOR</td>
<td>Circle end point monitoring factor</td>
</tr>
<tr>
<td>21100</td>
<td>ORIENTATION_IS_EULER</td>
<td>Angle definition for orientation programming</td>
</tr>
<tr>
<td>21110</td>
<td>X_AXIS_IN_OLD_X_Z_PLANE</td>
<td>Coordinate system for automatic Frame definition</td>
</tr>
<tr>
<td>21200</td>
<td>LIFTFAST_DIST</td>
<td>Traversing path for fast retraction from the contour</td>
</tr>
<tr>
<td>21210</td>
<td>SETINT_ASSIGN_FASTIN</td>
<td>HW assignment of the ext. NCK input byte for NC program interrupt:</td>
</tr>
<tr>
<td>21202</td>
<td>LIFTFAST_WITH_MIRROR</td>
<td>Lift fast with mirror</td>
</tr>
</tbody>
</table>

### Block search

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20128</td>
<td>COLLECT_TOOL_CHANGE</td>
<td>Tool change commands to the PLC after block search</td>
</tr>
<tr>
<td>22600</td>
<td>SERUPRO_SPEED_MODE</td>
<td>Velocity with block search type 5</td>
</tr>
<tr>
<td>22601</td>
<td>SERUPRO_SPEED_FACTOR</td>
<td>Velocity factor for block search type 5</td>
</tr>
<tr>
<td>22621</td>
<td>ENABLE_START_MODE_MASK_PRT</td>
<td>Switches MD22620: START_MODE_MASK_PRT for SERUPRO search run</td>
</tr>
<tr>
<td>22622</td>
<td>DISABLE_PLC_START</td>
<td>Allow part program start via PLC</td>
</tr>
<tr>
<td>22680</td>
<td>AUTO_IPTR_LOCK</td>
<td>Disable interrupt pointer</td>
</tr>
</tbody>
</table>

### Reset response

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20110</td>
<td>RESET_MODE_MASK</td>
<td>Initial setting after RESET / parts program end</td>
</tr>
<tr>
<td>20112</td>
<td>START_MODE_MASK</td>
<td>Basic setting for NC start after part program start</td>
</tr>
<tr>
<td>20118</td>
<td>GEOAX_CHANGE_RESET</td>
<td>Allow automatic geometry axis change</td>
</tr>
<tr>
<td>20120</td>
<td>TOOL_RESET_VALUE</td>
<td>Tool length compensation when powering-up (RESET / part program end)</td>
</tr>
<tr>
<td>20121</td>
<td>TOOL_PRESEL_RESET_VALUE</td>
<td>Preselected tool on RESET</td>
</tr>
<tr>
<td>20130</td>
<td>CUTTING_EDGE_RESET_VALUE</td>
<td>Tool cutting edge length compensation when powering-up (RESET / part program end)</td>
</tr>
<tr>
<td>20140</td>
<td>TRAFO_RESET_VALUE</td>
<td>Transformation data set when powering-up (RESET / part program end)</td>
</tr>
<tr>
<td>20150</td>
<td>GCODE_RESET_VALUES</td>
<td>Initial setting of the G groups</td>
</tr>
<tr>
<td>20152</td>
<td>GCODE_RESET_MODE</td>
<td>Reset behavior of G groups</td>
</tr>
</tbody>
</table>
### Auxiliary function settings

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC__ $</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>22000</td>
<td>AUXFU_ASSIGN_GROUP</td>
<td>Auxiliary function group</td>
</tr>
<tr>
<td>22010</td>
<td>AUXFU_ASSIGN_TYPE</td>
<td>Auxiliary function type</td>
</tr>
<tr>
<td>22020</td>
<td>AUXFU_ASSIGN_EXTENSION</td>
<td>Auxiliary function extension</td>
</tr>
<tr>
<td>22030</td>
<td>AUXFU_ASSIGN_VALUE</td>
<td>Auxiliary function value</td>
</tr>
<tr>
<td>22200</td>
<td>AUXFU_M_SYNC_TYPE</td>
<td>Output timing of M functions</td>
</tr>
<tr>
<td>22210</td>
<td>AUXFU_S_SYNC_TYPE</td>
<td>Output timing of S functions</td>
</tr>
<tr>
<td>22220</td>
<td>AUXFU_T_SYNC_TYPE</td>
<td>Output timing of T functions</td>
</tr>
<tr>
<td>22230</td>
<td>AUXFU_H_SYNC_TYPE</td>
<td>Output timing of H functions</td>
</tr>
<tr>
<td>22240</td>
<td>AUXFU_F_SYNC_TYPE</td>
<td>Output timing of F functions</td>
</tr>
<tr>
<td>22250</td>
<td>AUXFU_D_SYNC_TYPE</td>
<td>Output timing of D functions</td>
</tr>
<tr>
<td>22400</td>
<td>S_VALUES_ACTIVE_AFTER_RESET</td>
<td>S function active after RESET</td>
</tr>
<tr>
<td>22410</td>
<td>F_VALUES_ACTIVE_AFTER_RESET</td>
<td>F function active after RESET</td>
</tr>
<tr>
<td>22510</td>
<td>GCODE_GROUPS_TO_PLC</td>
<td>G codes that are output to the NCK/PLC interface on block change/RESET</td>
</tr>
<tr>
<td>22550</td>
<td>TOOL_CHANGE_MODE</td>
<td>New tool offset for M function</td>
</tr>
<tr>
<td>22560</td>
<td>TOOL_CHANGE_M_CODE</td>
<td>M function for tool change</td>
</tr>
</tbody>
</table>

### Memory settings

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC__ $</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>27900</td>
<td>REORG_LOG_LIMIT</td>
<td>Percentage of IPO buffer for log file enable</td>
</tr>
<tr>
<td>28000</td>
<td>MM_REORG_LOG_FILE_MEM</td>
<td>Memory size for REORG (DRAM)</td>
</tr>
<tr>
<td>28010</td>
<td>MM_NUM_REORG_LUD_MODULES</td>
<td>Number of blocks for local user variables for REORG</td>
</tr>
<tr>
<td>28020</td>
<td>MM_NUM_LUD_NAMES_TOTAL</td>
<td>Number of local user variables (DRAM)</td>
</tr>
<tr>
<td>28040</td>
<td>MM_LUD_VALUES_MEM</td>
<td>Memory size for local user variables (DRAM)</td>
</tr>
<tr>
<td>28050</td>
<td>MM_NUM_R_PARAM</td>
<td>Number of channelspecific R parameters (SRAM)</td>
</tr>
<tr>
<td>28060</td>
<td>MM_IPO_BUFFER_SIZE</td>
<td>Number of NC blocks in IPO buffer (DRAM)</td>
</tr>
<tr>
<td>28070</td>
<td>MM_NUM_BLOCKS_IN_PREP</td>
<td>Number of blocks for block preparation (DRAM)</td>
</tr>
<tr>
<td>28080</td>
<td>MM_NUM_USER_FRAMES</td>
<td>Number of settable Frames (SRAM)</td>
</tr>
<tr>
<td>28090</td>
<td>MM_NUM_CC_BLOCK_ELEMENTS</td>
<td>Number of block elements for compile cycles (DRAM)</td>
</tr>
<tr>
<td>28100</td>
<td>MM_NUM_CC_BLOCK_USER_MEM</td>
<td>Size of block memory for compile cycles (DRAM)</td>
</tr>
<tr>
<td>28400</td>
<td>MM_ABSBLOCK</td>
<td>Activating basis blocks with absolute values</td>
</tr>
<tr>
<td>28402</td>
<td>MM_ABSBLOCK_BUFFER_CONF</td>
<td>Dimension size of upload buffer</td>
</tr>
<tr>
<td>28500</td>
<td>MM_PREP_TASK_STACK_SIZE</td>
<td>Stack size of preparation task (DRAM)</td>
</tr>
</tbody>
</table>
Program runtime and workpiece counter

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>27860</td>
<td>PROCESSTIMER_MODE</td>
<td>Activate the runtime measurement</td>
</tr>
<tr>
<td>27880</td>
<td>PART_COUNTER</td>
<td>Activate the workpiece counter</td>
</tr>
<tr>
<td>27882</td>
<td>PART_COUNTER_MCODE[ ]</td>
<td>Workpiece counting via M command</td>
</tr>
</tbody>
</table>

9.19.1.3 Axis/spindlespecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>30465</td>
<td>AXIS_LANG_SUB_MASK</td>
<td>Substitution of NC language commands</td>
</tr>
<tr>
<td>30550</td>
<td>AXCONF.Assign_MASTER_CHAN</td>
<td>Reset position of channel for axis change</td>
</tr>
<tr>
<td>30600</td>
<td>FIX_POINT_POS</td>
<td>Fixed value positions of axes with G75</td>
</tr>
<tr>
<td>33100</td>
<td>COMPRESS_POS_TOL</td>
<td>Maximum deviation with compensation</td>
</tr>
</tbody>
</table>

9.19.2 Setting data

9.19.2.1 Channelspecific setting data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $SC__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>42000</td>
<td>THREAD_START_ANGLE</td>
<td>Start angle for thread</td>
</tr>
<tr>
<td>42010</td>
<td>THREAD_RAMP_DISP</td>
<td>Acceleration behavior of axis when thread cutting</td>
</tr>
<tr>
<td>42100</td>
<td>DRY_RUN_FEED</td>
<td>Dry run feedrate</td>
</tr>
<tr>
<td>42200</td>
<td>SINGLEBLOCK2_STOPRE</td>
<td>Activate debug mode for SBL2</td>
</tr>
<tr>
<td>42444</td>
<td>TARGET_BLOCK_INCR_PROG</td>
<td>Continuation mode after block search with calculation</td>
</tr>
<tr>
<td>42700</td>
<td>EXT_PROG_PATH</td>
<td>Program path for external subroutine call EXTCALL</td>
</tr>
<tr>
<td>42750</td>
<td>ABSBLOCK_ENABLE</td>
<td>Enable basic block display</td>
</tr>
<tr>
<td>42990</td>
<td>MAX_BLOCKS_IN_IPOBUFFER</td>
<td>Maximum number of blocks in the interpolation buffer</td>
</tr>
</tbody>
</table>

9.19.3 Signals

9.19.3.1 Signals to NC

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergency stop</td>
<td>DB10.DBX56.1</td>
<td>DB2600.DBX0.1</td>
</tr>
</tbody>
</table>
### 9.19.3.2 Signals to mode group

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOMATIC mode</td>
<td>DB11.DBX0.0</td>
<td>DB3000.DBX0.0</td>
</tr>
<tr>
<td>MDA mode</td>
<td>DB11.DBX0.1</td>
<td>DB3000.DBX0.1</td>
</tr>
<tr>
<td>JOG mode</td>
<td>DB11.DBX0.2</td>
<td>DB3000.DBX0.2</td>
</tr>
<tr>
<td>Mode change disable</td>
<td>DB11.DBX0.4</td>
<td>DB3000.DBX0.4</td>
</tr>
<tr>
<td>Mode group stop</td>
<td>DB11.DBX0.5</td>
<td>DB3000.DBX0.5</td>
</tr>
<tr>
<td>Mode group stop axes plus spindles</td>
<td>DB11.DBX0.6</td>
<td>DB3000.DBX0.6</td>
</tr>
<tr>
<td>Mode group reset</td>
<td>DB11.DBX0.7</td>
<td>DB3000.DBX0.7</td>
</tr>
<tr>
<td>Machine function teach in</td>
<td>DB11.DBX1.0</td>
<td>DB3000.DBX1.0</td>
</tr>
<tr>
<td>Machine function REPOS</td>
<td>DB11.DBX1.1</td>
<td>-</td>
</tr>
<tr>
<td>Machine function REF</td>
<td>DB11.DBX1.2</td>
<td>DB3000.DBX1.2</td>
</tr>
</tbody>
</table>

### 9.19.3.3 Signals to NC

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected mode AUTOMATIC</td>
<td>DB11.DBX4.0</td>
<td>-</td>
</tr>
<tr>
<td>Selected mode MDA</td>
<td>DB11.DBX4.1</td>
<td>-</td>
</tr>
<tr>
<td>Selected JOG mode</td>
<td>DB11.DBX4.2</td>
<td>-</td>
</tr>
<tr>
<td>Selected machine function teach in</td>
<td>DB11.DBX5.0</td>
<td>-</td>
</tr>
<tr>
<td>Selected machine function REPOS</td>
<td>DB11.DBX5.1</td>
<td>-</td>
</tr>
<tr>
<td>Selected machine function REF</td>
<td>DB11.DBX5.2</td>
<td>-</td>
</tr>
<tr>
<td>Active mode AUTOMATIC</td>
<td>DB11.DBX6.0</td>
<td>DB3100.DBX0.0</td>
</tr>
<tr>
<td>Active mode MDA</td>
<td>DB11.DBX6.1</td>
<td>DB3100.DBX0.1</td>
</tr>
<tr>
<td>Active mode JOG</td>
<td>DB11.DBX6.2</td>
<td>DB3100.DBX0.2</td>
</tr>
<tr>
<td>Mode group ready</td>
<td>DB11.DBX6.3</td>
<td>DB3100.DBX0.3</td>
</tr>
<tr>
<td>Mode group reset performed</td>
<td>DB11.DBX6.4</td>
<td>DB3100.DBX0.4</td>
</tr>
<tr>
<td>NC internal JOG active</td>
<td>DB11.DBX6.5</td>
<td>-</td>
</tr>
<tr>
<td>All channels in the reset state</td>
<td>DB11.DBX6.7</td>
<td>DB3100.DBX0.7</td>
</tr>
<tr>
<td>Active machine function teach in</td>
<td>DB11.DBX7.0</td>
<td>DB3100.DBX1.0</td>
</tr>
<tr>
<td>Active machine function REPOS</td>
<td>DB11.DBX7.1</td>
<td>-</td>
</tr>
<tr>
<td>Active machine function REF</td>
<td>DB11.DBX7.2</td>
<td>DB3100.DBX1.2</td>
</tr>
</tbody>
</table>

### 9.19.3.4 Signals to channel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate DRF</td>
<td>DB21, ... .DBX0.3</td>
<td>DB320x.DBX0.3</td>
</tr>
<tr>
<td>Activate single block</td>
<td>DB21, ... .DBX0.4</td>
<td>DB320x.DBX0.4</td>
</tr>
<tr>
<td>Activate M01</td>
<td>DB21, ... .DBX0.5</td>
<td>DB320x.DBX0.5</td>
</tr>
<tr>
<td>Activate dry run feedrate</td>
<td>DB21, ... .DBX0.6</td>
<td>DB320x.DBX0.6</td>
</tr>
</tbody>
</table>
## 9.19 | Data lists

### K1: Mode group, channel, program operation, reset response

#### 9.19.3.5 Signals from channel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC action completed</td>
<td>DB21, ... .DBX1.6</td>
<td>-</td>
</tr>
<tr>
<td>Activate program test</td>
<td>DB21, ... .DBX1.7</td>
<td>DB320x.DBX1.7</td>
</tr>
<tr>
<td>Skip block levels: /0 to /7</td>
<td>DB21, ... .DBX2.0-7</td>
<td>DB320x.DBX2.0-7</td>
</tr>
<tr>
<td>Read-in disable</td>
<td>DB21, ... .DBX6.1</td>
<td>DB320x.DBX6.1</td>
</tr>
<tr>
<td>Program level abort</td>
<td>DB21, ... .DBX6.4</td>
<td>DB320x.DBX6.4</td>
</tr>
<tr>
<td>NC start disable</td>
<td>DB21, ... .DBX7.0</td>
<td>DB320x.DBX7.0</td>
</tr>
<tr>
<td>NC Stop at block limit</td>
<td>DB21, ... .DBX7.1</td>
<td>DB320x.DBX7.1</td>
</tr>
<tr>
<td>NC stop</td>
<td>DB21, ... .DBX7.2</td>
<td>DB320x.DBX7.2</td>
</tr>
<tr>
<td>NC Stop axes plus spindles</td>
<td>DB21, ... .DBX7.3</td>
<td>DB320x.DBX7.3</td>
</tr>
<tr>
<td>NC start disable</td>
<td>DB21, ... .DBX7.4</td>
<td>DB320x.DBX7.4</td>
</tr>
<tr>
<td>Reset</td>
<td>DB21, ... .DBX7.7</td>
<td>DB330x.DBX3.7</td>
</tr>
<tr>
<td>REPOSPATHMODE</td>
<td>DB21, ... .DBX31.0-2</td>
<td>-</td>
</tr>
<tr>
<td>REPOS MODE EDGE</td>
<td>DB21, ... .DBX31.4</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRF selected</td>
<td>DB21, ... .DBX24.3</td>
<td>DB170x.DBX0.3</td>
</tr>
<tr>
<td>Select NCK associated M01</td>
<td>DB21, ... .DBX24.4</td>
<td>-</td>
</tr>
<tr>
<td>M01 selected</td>
<td>DB21, ... .DBX24.5</td>
<td>DB170x.DBX0.5</td>
</tr>
<tr>
<td>Dry run feedrate selected</td>
<td>DB21, ... .DBX24.6</td>
<td>DB170x.DBX0.6</td>
</tr>
<tr>
<td>Feedrate override for rapid traverse selected</td>
<td>DB21, ... .DBX25.3</td>
<td>DB170x.DBX1.3</td>
</tr>
<tr>
<td>REPOS MODE EDGE</td>
<td>DB21, ... .DBX25.4</td>
<td>-</td>
</tr>
<tr>
<td>Program test selected</td>
<td>DB21, ... .DBX25.7</td>
<td>DB170x.DBX1.7</td>
</tr>
<tr>
<td>Skip block selected /0 - /7</td>
<td>DB21, ... .DBX26.0-7</td>
<td>DB170x.DBX2.0-7</td>
</tr>
<tr>
<td>Skip block selected /8</td>
<td>DB21, ... .DBX27.0</td>
<td>DB170x.DBX3.0</td>
</tr>
<tr>
<td>Skip block selected /9</td>
<td>DB21, ... .DBX27.1</td>
<td>DB170x.DBX3.1</td>
</tr>
<tr>
<td>REPOSPATHMODE 0 - 2</td>
<td>DB21, ... .DBX31.0-2</td>
<td>-</td>
</tr>
<tr>
<td>REPOS MODE EDGE</td>
<td>DB21, ... .DBX31.4</td>
<td>-</td>
</tr>
<tr>
<td>Skip block active /8</td>
<td>DB21, ... .DBX31.6</td>
<td>DB320x.DBX15.6</td>
</tr>
<tr>
<td>Skip block active /9</td>
<td>DB21, ... .DBX31.7</td>
<td>DB320x.DBX15.7</td>
</tr>
<tr>
<td>Execution from external active</td>
<td>DB21, ... .DBX32.0</td>
<td>DB330x.DBX0.0</td>
</tr>
<tr>
<td>Action block active</td>
<td>DB21, ... .DBX32.3</td>
<td>DB330x.DBX0.3</td>
</tr>
<tr>
<td>Approach block active</td>
<td>DB21, ... .DBX32.4</td>
<td>DB330x.DBX0.4</td>
</tr>
<tr>
<td>M0/M1 active</td>
<td>DB21, ... .DBX32.5</td>
<td>DB330x.DBX0.5</td>
</tr>
<tr>
<td>Last action block active</td>
<td>DB21, ... .DBX32.6</td>
<td>DB330x.DBX0.6</td>
</tr>
<tr>
<td>Block search active</td>
<td>DB21, ... .DBX33.4</td>
<td>DB330x.DBX1.4</td>
</tr>
<tr>
<td>M02/M30 active</td>
<td>DB21, ... .DBX33.5</td>
<td>DB330x.DBX1.5</td>
</tr>
<tr>
<td>Transformation active</td>
<td>DB21, ... .DBX33.6</td>
<td>DB330x.DBX1.6</td>
</tr>
</tbody>
</table>
### 9.19.3.6 Signals to NC

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPOS DELAY</td>
<td>DB31, ..., DBX10.0</td>
<td>-</td>
</tr>
</tbody>
</table>

### 9.19.3.7 Signals from axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPOS offset</td>
<td>DB31, ..., DBX70.0</td>
<td>-</td>
</tr>
<tr>
<td>REPOS offset valid</td>
<td>DB31, ..., DBX70.1</td>
<td>-</td>
</tr>
<tr>
<td>REPOS Delay Ackn</td>
<td>DB31, ..., DBX70.2</td>
<td>-</td>
</tr>
<tr>
<td>REPOS DELAY</td>
<td>DB31, ..., DBX72.0</td>
<td>-</td>
</tr>
<tr>
<td>Path axis</td>
<td>DB31, ..., DBX76.4</td>
<td>DB390x.DBX1002.4</td>
</tr>
</tbody>
</table>
9.19 Data lists
10.1 Brief description

10.1.1 Axes

Machine axes
Machine axes are the axes that actually exist on a machine tool.

Channel axes
Every geometry axis and every special axis is assigned to a channel and, therefore, a channel axis. Geometry axes and additional axes are always traversed in “their” channel.

Geometry axes
The three geometry axes always make up a fictitious rectangular coordinate system, the basic coordinate system (BCS).

By using FRAMES (offset, rotation, scaling, mirroring), it is possible to image geometry axes of the workpiece coordinate system (WCS) on the BCS.

Special axes
In contrast to geometry axes, no geometrical relationship is defined between the special axes.

Path axes
Path axes are interpolated together (all the path axes of a channel have a common path interpolator).

All the path axes of one channel have the same acceleration phase, constant travel phase and delay phase.

Positioning axes
Positioning axes are interpolated separately (each positioning axis has its own axis interpolator). Each positioning axis has its own feedrate and acceleration characteristic.

Synchronized axes
Synchronous axes are interpolated together with path axes (all path axes and synchronous axes of one channel have a common path interpolator).
All path axes and all synchronous axes of a channel have the same acceleration phase, constant travel phase and deceleration phase.

Axis configuration

The machine data below are used to assign the geometry axes, special axes, channel axes and machine axes as well as the names of the individual axis types:

- MD20050 $MC_AXCONF_GEOAX_ASGN_TAB (assignment of geometry axis to channel axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB (name of the geometry axis in the channel)
- MD20070 $MC_AXCONF_MACHAX_USED (machine axis number valid in channel)
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB (name of the channel axis in the channel)
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB (machine axis name)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX (assignment of spindle to machine axis)

Replaceable geometry axes

The "Replaceable geometry axes" function allows the geometry axes in a grouping to be replaced by other channel axes.

Axes that are initially configured as synchronous special axes in a channel can replace any selected geometry axis in response to a program command.

Link axis

Link axes are axes, which are physically connected to another NCU and whose position is controlled from this NCU. Link axes can be assigned dynamically to channels of another NCU. Link axes are not local axes from the perspective of a particular NCU.

The axis container concept is used for the dynamic modification of the assignment to an NCU. Axis replacement with GET and RELEASE from the part program is not available for link axes across NCU boundaries.

The link axes are described in

References:
Function Manual, Extended Functions; Several Operator Panels on Multiple NCUs, Distributed Systems (B3)

Axis container

An axis container is a circular buffer data structure, in which local axes and/or link axes are assigned to channels. The entries in the circular buffer can be shifted cyclically.

In addition to the direct reference to local axes or link axes, the link axis configuration in the logical machine axis image also allows references to axis containers.

This type of reference consists of:
- Axis container number
- A slot (circular buffer location within the corresponding container)
The entry in a circular buffer location contains:

- A local axis
  or
- A link axis

The axis container function is described in

References:
Function Manual, Extended Functions; Several Operator Panels on Multiple NCUs, Distributed Systems (B3)

10.1.2 Coordinate systems

MCS

The machine coordinate system (MCS) has the following properties:

- It is defined by the machine axes.
- The machine axes can be perpendicular to each other to form Cartesian system or arranged in any other way.
- The names of the machine axes can be defined.
- The machine axes can be linear or rotary axes.

BCS

The basic coordinates system (BKS) has the following properties:

- The geometry axes form a perpendicular Cartesian coordinate system.
- The BCS is derived from a kinematic transformation of the MCS.

BZS

The basic zero system (BZS) is the basic coordinate system with a basic offset.

SZS

The settable zero system (SZS) is the workpiece coordinate system with a programmable frame from the viewpoint of the WCS. The workpiece zero is defined by the settable frames G54 to G599.

WCS

The workpiece coordinate system (WCS) has the following properties:

- In the workpiece coordinate system all the axes coordinates are programmed (parts program).
- It is made up of geometry axes and special axes.
• Geometry axes always form a perpendicular Cartesian coordinate system
• Special axes form a coordinate system without any geometrical relation between the special axes.
• The names of the geometry axes and special axes can be defined.
• The workpiece coordinate system can be translated, rotated, scaled or mirrored with FRAMES (TRANS, ROT, SCALE, MIRROR). Multiple translations, rotational movements, etc., are also possible.

Zero offset external

The zero offset external has the following properties:
• At a time defined in the PLC, a predefined additional zero offset between the basic and the workpiece coordinate systems is activated.
• The magnitudes of the offsets can be set by the following for each of the axes involved:
  – PLC
  – Operator Panel
  – Part program
• Activated offsets take effect at the instant the first motion block of the relevant axes is processed after offset activation. The offsets are superimposed on the programmed path (no interpolation).

The velocity, at which the zero offset external is applied, is as follows:
Programmed $F$ value plus $+1/2$ JOG velocity
The zero offset external is traversed at the end of $G0$ blocks.
• The activated offsets are retained after RESET and end of program.
• After POWER ON, the last active offset is still stored in the control but must be reactivated by the PLC.

10.1.3 Frames

A frame is a closed calculation rule (algorithm) that translates one Cartesian coordinate system into another.

Frame components

A frame consists of the following components:

<table>
<thead>
<tr>
<th>Frame components</th>
<th>Programmable with:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset</td>
<td></td>
</tr>
<tr>
<td>Coarse offset</td>
<td>TRANS</td>
</tr>
<tr>
<td></td>
<td>ATRANS (additive translation component)</td>
</tr>
<tr>
<td></td>
<td>CTRANS (work offset for multiple axes)</td>
</tr>
<tr>
<td></td>
<td>G58 (axial work offset)</td>
</tr>
<tr>
<td>Fine offset</td>
<td>CFINE</td>
</tr>
<tr>
<td></td>
<td>G59 (axial work offset)</td>
</tr>
</tbody>
</table>
Coarse and fine offsets

As the assignment of machine axes to channel axes and, in particular, to geometry axes, can be different in all channels, there is consequently no unique cross-channel geometric relationship between the channel axes. Therefore, only offset, scaling and mirroring is possible for NCU-global frames. Rotations are not possible.

G58: Absolute axis-specific programmable work offset (coarse offset)
The absolute component of the translatory offset (coarse offset) is programmed axis-specifically with G58. The additive component of the translatory offset (fine offset) is retained.

G59: Additive axis-specific programmable work offset (fine offset)
The additive component of the translatory offset (fine offset) is programmed axis-specifically with G59. The absolute component of the translatory offset (coarse offset) is retained.

G59 can only be used when the fine offset has been released:
- MD18600 $MN_MM_FRAME_FINE_TRANS = TRUE
- MD24000 $MC_FRAME_ADD_COMPONENTS = TRUE
Rotation

Orientations in space are defined via frame rotations as follows:

- Rotation with `ROT` defines the individual rotations for all geometry axes.
- Solid angles with `ROTS`, `AROTS`, `CROTS` define the orientation of a plane in space.
- Frame rotation with `TOFRAME` defines a frame with a Z axis pointing in the tool direction.

Scaling

`SCALE` is used to program the programmable scale factors for all geometry axes and special axes.

If a new scaling is to be based on a previous scaling, rotation, translation or mirroring, then `ASCALE` must be programmed.

Mirroring

The following machine data is used to set how the mirroring is to be performed:

```
MD10610 $MN_MIRROR_REF_AX
```

Concatenation

Frames and frame components can be combined using the concatenation operator `:`. The channel-specific active frame `$P_ACTFRAME` results, for example, from the chaining of all active frames of the channel:

```
$P_ACTFRAME = $P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME :
$P_ISO1FRAME : $P_ISO2FRAME : $P_ISO3FRAME :
$P_ACTBF : $P_IFRAME : $P_GFRAME :
$P_TOOLFRAME : $P_WPFRAME : $P_TRAFRAME :
$P_PF : $P_ISO4FRAME : $P_CYCFRAME
```

Supplementary conditions

**Incremental dimensioning G91**

Incremental programming with `G91` is defined such that the offset value is traversed additively to the incrementally programmed value when a work offset is selected.

The behavior depends on the setting in the setting data:

```
SD42440 $SC_FRAME_OFFSET_INCR_PROG (work offsets in frames)
```

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Work offset is applied on FRAME and incremental programming of an axis (= default setting).</td>
</tr>
<tr>
<td>0</td>
<td>Only the programmed path is traversed.</td>
</tr>
</tbody>
</table>
Consistency

When writing, reading and activating frames, e.g. using channel coordination, the user is solely responsible for achieving consistent behavior within the channels. Cross-channel activation of frames is not supported.

10.2 Axes

10.2.1 Overview

Figure 10-2  Relationship between geometry axes, special axes and machine axes
Figure 10-3   Local and external machine axes (link axes)
10.2.2 Machine axes

Meaning

Machine axes are the axes that actually exist on a machine tool.

Application

The following can be machine axes:

- Geometry axes X, Y, Z
- Orientation axes A, B, C
- Loader axes
- Tool turrets

Figure 10-4 Machine axes X, Y, Z, B, S on a Cartesian machine
• Axes for tool magazine
• Axes for automatic tool changer
• Spindle sleeves
• Axes for pallet changers
• Etc.

10.2.3 Channel axes

Meaning
Each geometry axis and each special axis is assigned to a channel. Geometry axes and additional axes are always traversed in "their" channel.

10.2.4 Geometry axes

Meaning
The three geometry axes always make up a fictitious rectangular coordinate system.
By using FRAMES (offset, rotation, scaling, mirroring), it is possible to image geometry axes of the workpiece coordinate system (WCS) on the BCS.

Application
Geometry axes are used to program the workpiece geometry (the contour).
Plane selection \texttt{G17, G18 and G19} (DIN 66217) always refers to the three geometry axes. That is why it is advantageous to name the three geometry axes X, Y and Z.

10.2.5 Special axes

Significance
In contrast to geometry axes, no geometrical relationship is defined between the special axes.

Note
Geometry axes have an exactly defined relationship in the form of a rightangled coordinate system.
Special axes are part of the basic coordinate system (BCS). With FRAMES (translation, scaling, mirroring), special axes of the workpiece coordinate system can be mapped on the basic coordinate system.

**Application**

Typical special axes are:
- Rotary axes
- Machine tool axes
- Tool revolver axes
- Loader axes

### 10.2.6 Path axes

**Meaning**

Path axes are interpolated together (all the path axes of a channel have a common path interpolator).

All the path axes of one channel have the same acceleration phase, constant travel phase and delay phase.

The feedrate programmed under address $F$ (path feedrate) applies to all the path axes programmed in a block, with the following exceptions:

- An axis has been programmed that has been defined as having no control over the path velocity with instruction $\text{FGROUP}$.
- Axes programmed with instructions $\text{POS}$ or $\text{POSA}$ have an individual feedrate setting (axis interpolator).

**Application**

Path axes are used to machine the workpiece with the programmed contour.

### 10.2.7 Positioning axes

**Meaning**

Positioning axes are interpolated separately (each positioning axis has its own axis interpolator). Each positioning axis has its own feedrate and acceleration characteristic. Positioning axes can be programmed in addition to path axes (even in the same block). Path axis interpolation (path interpolator) is not affected by the positioning axes. Path axes and the individual positioning axes do not necessarily reach their block end points at the same time.
Instructions **POS** and **POSA** are used to program positioning axes and define block change criteria:

- **POS**
  Block change takes place when the path axes and positioning axes have reached their block end points.

- **POSA**
  Block change takes place when the path axes have reached their end of block position. Positioning axes continue to traverse beyond block limits to their block end point.

Concurrent positioning axes differ from positioning axes in that they:

- Only receive their block end points from the PLC
- Can be started at any time (not at block limits)
- Do not affect the execution of current part programs.

**Application**

Typical positioning axes are:

- Loaders for moving workpieces away from machine
- Tool magazine/turret

**Reference**

For further information, see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 821)" and "S1: Spindles (Page 1199)".

**References:**

- Function Manual, Extended Functions; Positioning Axes (P2)
- Function Manual, Special Functions; Gantry Axes (G1)
- Function Manual, Special Functions; Axis Couplings and ESR (M3)
- Function Manual, Extended Functions; Oscillation (P5)
- Function Manual, Synchronized Actions

### 10.2.8 Main axes

**Meaning**

A main axis is an axis that is interpolated by the main run.
This interpolation can be started as follows:

- From synchronized actions (as command axes due to an event via block-related, modal or static synchronized actions)
- From the PLC via special function blocks in the basic PLC program (named as a concurrent positioning axis or a PLC axis)
- Via the setting data or from the part program (as an asynchronous or block-synchronous oscillating axis)

**Channel control**

An axis interpolated by the main axis reacts in terms of:

- NC stop
- Alarm handling
- Program control
- End of program
- RESET

**Note**

The response at the end of the program varies. The axis movement need not always be completed and, therefore, may carry on beyond the end of the program.

**Application**

Certain axes in the main run can be decoupled at the channel response triggered by the NC program sequence and controlled from the PLC. These axes are also interpolated in the main run and respond independently for the channel and program sequence.

A PLC-controlled axis can then be controlled independently by the NC. This concerns the following actions:

- The sequence for canceling the axis (equivalent to delete distancetogo)
- Stopping or interrupting the axis
- Continuing the axis (continue sequence of motion)
- Resetting the axis to its basic status

### 10.2.9 Synchronized axes

**Meaning**

Synchronous axes are components of the path axes, which are not referenced in order to calculate the tool path velocity. They are interpolated together with path axes (all path axes and synchronous axes of one channel have a common path interpolator).
All path axes and all synchronous axes of a channel have the same acceleration phase, constant travel phase and deceleration phase.

The feedrate (path feedrate) programmed under address $F$ applies to all the path axes programmed in a block but not to the synchronous axes.

Synchronous axes take the same time to cover the programmed path as the path axes.

**FGROUP command**

The command **FGROUP** specifies whether the axis is a feed-defining **path axis** (used to calculate the path velocity) or a **synchronous axis** (not used to calculate the path velocity).

**Example**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N05 G00 G94 G90 M3 S1000 X0 Y0 Z0 ;</td>
<td>Axes X/Y are path axes, Z is a synchronous axis.</td>
</tr>
<tr>
<td>N10 FGROUP(X,Y)</td>
<td>Progr. feedrate 1000 mm/min. Feedrate of axis X = 707 mm/min. Feedrate of axis Y = 707 mm/min.</td>
</tr>
<tr>
<td>N20 G01 X100 Y100 P1000</td>
<td>Progr. Feedrate 1000 mm/min Feedrate of Axis X = 707 mm/min Feedrate of Axis Y = 700 mm/min, because only half the distance is to be traversed.</td>
</tr>
<tr>
<td>N30 FGROUP (X)</td>
<td>Axis X is a path axis, axis Y is a synchronous axis</td>
</tr>
<tr>
<td>N20 X200 Y150</td>
<td>Progr. Feedrate 1000 mm/min Feedrate of Axis X = 1000 mm/min Feedrate of Axis Y is set to 500 mm/min, because only half the distance is to be traversed.</td>
</tr>
</tbody>
</table>

**Note**

The channel axis name must be used for the **FGROUP** command.

This is defined by the machine data:

MD20080 $MC_AXCONF_CHANAX_NAME_TAB (name of the channel axis in the channel)

**Application**

In the case of helical interpolation **FGROUP** can be programmed to determine whether:

- The programmed feedrate should be valid on the path (all three programmed axes are path axes)
- The programmed feedrate should be valid on the circle (two axes are path axes and the infeed axis is a synchronous axis)
10.2.10 **Axis configuration**

Assigning geometry, special, channel and machine axes.

---

**NCU 1**

Geometry axis names

\[ \text{MD20060 } \text{SMC\_AXCONF\_GEOAX\_NAME\_TAB[k][n]} \]

- 1st geometry axis \[ [0] = "X" \]
- 2nd geometry axis \[ [1] = "Y" \]
- 3rd geometry axis \[ [2] = "Z" \]

Used channel axes

\[ \text{MD20050 } \text{SMC\_AXCONF\_GEOAX\_ASSIGN\_TAB[k][n]} \]

- 1st geometry axis
  - Channel 1
    - \[ [0] = 1 \]
    - \[ [1] = 2 \]
    - \[ [2] = 3 \]
- 2nd geometry axis
  - Channel 1
    - \[ [0] = 1 \]
    - \[ [1] = 2 \]
    - \[ [2] = 3 \]

**Channel axis name**

\[ \text{MD20080 } \text{SMC\_AXCONF\_CHANAX\_NAME\_TAB[k][n]} \]

- 1st channel axis \[ [0] = "X" \]
- 2nd channel axis \[ [1] = "Y" \]
- 3rd channel axis \[ [2] = "Z" \]
- 4th channel axis \[ [3] = "A" \]
- 5th channel axis \[ [4] = "B" \]
- 6th channel axis \[ [5] = "U" \]
- 7th channel axis \[ [6] = "V" \]
- 8th channel axis \[ [7] = "W" \]

**Used machine axes**

\[ \text{MD20070 } \text{SMC\_AXCONF\_MACHAX\_USED[k][n]} \]

- 1. Kanalache
  - Channel 1
    - \[ [0] = 1 \]
    - \[ [1] = 2 \]
    - \[ [2] = 3 \]
- 2. Kanalache
  - Channel 1
    - \[ [3] = 4 \]
- 3. Kanalache
  - Channel 1
    - \[ [4] = 5 \]
- 4. Kanalache
  - Channel 1
    - \[ [5] = 6 \]
- 5. Kanalache
  - Channel 1
    - \[ [6] = 7 \]
- 6. Kanalache
  - Channel 1
    - \[ [7] = 0 \]

**Logical machine axis image**

\[ \text{MD10002 } \text{SMC\_AXCONF\_LOGIC\_MACHAX\_TAB[n]} \]

- \[ AX1 = ^{AX2}^{AX3}^{AX4} \]
- \[ AX6 = ^{AX7}^{AX8} \]
- \[ CT1 = ^{CT2}^{CT3} \]
- \[ CT1 = ^{CT2}^{CT3} \]

**Local machine axes**

\[ \text{MD10000 } \text{SMC\_AXCONF\_MACHAX\_NAME\_TAB[n]} \]

- 1st machine axis \[ [0] = "AX1" \]
- 2nd machine axis \[ [1] = "AX2" \]
- 3rd machine axis \[ [2] = "AX3" \]
- 4th machine axis \[ [3] = "AX4" \]
- 5th machine axis \[ [4] = "AX5" \]
- 6th machine axis \[ [5] = "AX6" \]
- 7th machine axis \[ [6] = "AX7" \]
- 8th machine axis \[ [7] = "AX8" \]

**Axis container names**

\[ \text{MD12750 } \text{SMC\_AXCT\_NAME\_TAB[n]} \]

- \[ [0] = "CT1" \]
- \[ [1] = "CT2" \]
- \[ [2] = "CT3" \]
- \[ [3] = "CT4" \]

**Container-link axis**

Machine axis AXm on NCU n, with \( n \leftrightarrow \) "own NCU"

**Link axis**

Machine axis AXn on NCU 2
Special features

- Leading zeros for user-defined axis names are ignored:
  \( \text{MD10000} \cdot \$\text{MN AXCONF MACHAX NAME TAB}[0] = X01 \) corresponds to X1

- The geometry axes must be assigned to the channel axes in ascending order without any gaps.

- All channel axes that are not geometry axes are special axes.

Channel axis gaps

Normally, each channel axis defined in machine data MD20080
\( \$\text{MC AXCONF CHANAX NAME TAB} \) must be assigned a machine axis.

In order to simplify commissioning series of machines with a different number of machine axes,
channel axes may also be defined, which are not assigned to any machine axis. As a result,
gaps can occur in the numbering sequence of the channel axes.

Any channel axis gaps must be explicitly enabled:

\( \text{MD11640} \cdot \$\text{MN ENABLE_CHAN AX GAP} = 1 \)

Without being enabled, a value of 0 in machine data:

\( \text{MD20070} \cdot \$\text{MC AXCONF MACHAX USED} \)
ends the assignment of possibly existing additional machine axes to channel axes.

References:
Function Manual, Extended Functions; Several Operator Panels on Multiple NCUs, Distributed Systems (B3)

Note

Channel axes without assigned machine axes (channel axis gaps) are, regarding the number
and indexing of the channel axes, treated just like normal channel axes with associated
machine axes.

If a channel axis without assigned machine axis (channel axis gap) is defined as geometry
axis, then this is rejected without an alarm.
Example: Channel axis gap

Channel axis B is not assigned a machine axis in the following example.

![Diagram showing Channel axis configuration with channel axis gap](image)

Figure 10-6  Axis configuration with channel axis gap (excerpt)

Special situations: Channel axis gaps

Regarding channel axis gaps, the following also have to be taken into account:

- Channel axes without assigned machine axes (channel axis gaps) are, regarding the number and indexing of the channel axes, treated just like normal channel axes with associated machine axes.

- If a channel axis without assigned machine axis (channel axis gap) is defined as geometry axis, then this is rejected without an alarm.

10.2.11 Link axes

Meaning

A link axis is a machine axis that is not on the NCU from which it is traversed. The name of a local machine axis is not entered in the machine data for the logical machine axis image of the traversing NCU, but the NCU and machine axis name of the NCU to which it is physically connected.

As an example, machine axis AX1 of NCU2 should be traversed from NCU1:

- NCU1: MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB[n] = NC2_AX1

Requirement

The NCUs involved must be connected using link communication as a requirement for using link axes. The link axes and link communication functions are described in detail in:
10.3 Zeros and reference points

10.3.1 Reference points in working space

Zeros and reference points

The neutral position of the machine is obtained from the coordinate axes and the constructive characteristics of the machine. The zero of the coordinate system is obtained by defining a suitable reference point on the machine in its neutral position.

The position of the coordinate systems (MCS, BCS, BZS, SZS, WCS) is determined by means of zeros.

<table>
<thead>
<tr>
<th>Zero points</th>
<th>Reference points</th>
</tr>
</thead>
<tbody>
<tr>
<td>M = Machine zero</td>
<td>R = Reference point</td>
</tr>
<tr>
<td>W = Workpiece zero</td>
<td>T = Toolholder reference point</td>
</tr>
</tbody>
</table>

Machine zero M

The machine zero M defines the machine coordinate system MCS. All other reference points refer to the machine zero.

Workpiece zero W

The workpiece zero W defines the workpiece coordinate system in relation to the machine zero M. The programmed part-program blocks are executed in the workpiece coordinate system WCS.

Reference point R

The position of the reference point R is defined by cam switches. Reference point R calibrates the position measuring system.

With incremental encoders, the reference point must be approached every time the control power is switched on. The control can only then work with the measuring system and transfer all position values to the coordinate systems.
Toolholder reference point T

The toolholder reference point T is located on the toolholder locator. By entering the tool lengths, the control calculates the distance between the tool tip (TCP Tool Center Position) and the toolholder reference point.

Example: Zeros and reference points on a turning machine
10.3.2 Position of coordinate systems and reference points

Control POWER ON

For incremental measuring probes, the reference point must be approached each time the control is activated so that the control can transfer all position values to the coordinate system.

Figure 10-7 Position of coordinate systems by machine zero M and workpiece zero W

Figure 10-8 Position of reference point in relation to machine zero
10.4 Coordinate systems

10.4.1 Overview

Definitions

DIN 66217 stipulates that machine tools must use right-angled, rectangular (Cartesian) coordinate systems. The positive directions of the coordinate axes are determined using the "Right Hand Rule". The coordinate system is related to the workpiece and programming takes place independently of whether the tool or the workpiece is being traversed. When programming, it is always assumed that the tool traverses relative to the coordinate system of the workpiece, which is intended to be stationary.

Figure 10-9 Right-hand rule

Figure 10-10 Clockwise, rectangular Cartesian coordinate system
### Coordinate systems

The following coordinate systems are defined for machine tools:

<table>
<thead>
<tr>
<th>Coordinate system</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Coordinat System</td>
<td>MCS</td>
</tr>
<tr>
<td>Basic Coordinate System</td>
<td>BCS</td>
</tr>
<tr>
<td>Basic Zero System</td>
<td>BZS</td>
</tr>
<tr>
<td>Settable Zero System</td>
<td>SZS</td>
</tr>
<tr>
<td>Workpiece Coordinate System</td>
<td>WCS</td>
</tr>
</tbody>
</table>
Figure 10-11 Interrelationships between coordinate systems
Starting from the machine coordinate system, the various coordinate systems are defined using kinematic transformation and frames:

- **MCS ⇒ BCS**: Kinematic transformation
  If kinematic transformation is not active, then the BCS is the same as the MCS.
- **BCS ⇒ BZS**: Basic frame
- **BZS ⇒ SZS**: Activated adjustable FRAME G54...G599
- **SZS ⇒ WCS**: Programmable FRAME
- The program to machine the workpiece is written in WCS.
10.4.2 Machine coordinate system (MCS)

Machine coordinate system (MCS)

The machine coordinate system (MCS) is made up of all physically available machine axes.

Figure 10-12 MCS with machine axes X, Y, Z, B, C (5-axis milling machine)
Axial preset offset

The reference point of the control in the machine coordinate system (machine zero) can be reset via the “Preset offset (PRESETON)” function.

<table>
<thead>
<tr>
<th>CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Loss of the encoder adjustment</strong></td>
</tr>
<tr>
<td>After a preset offset, the appropriate machine axis is in the &quot;Not referenced&quot; state! This means that when using absolute encoders, the encoder adjustment is lost and must be performed again (e.g. by calibration with a laser interferometer). The use of PRESETON in combination with absolute encoders is therefore not recommended.</td>
</tr>
</tbody>
</table>

**Note**

We recommend that the function is only used for machine axes that do not require a reference point.

In order to restore the original machine coordinate system, the machine axis must be re-referenced, e.g. with G74 (reference point approach).

The machine axes are not moved with the preset offset.

**References**

- Programming Manual, Fundamentals
  Section: "Supplementary commands" > "Reference point approach (G74)"
- Programming Manual, Job Planning
  Section: "Coordinate transformations (FRAMES)" > "Preset offset (PRESETON)"

**10.4.2.1 Actual value setting with loss of the referencing status (PRESETON)**

**Function**

The PRESETON() procedure sets a new actual value for one or more axes in the machine coordinate system (MCS). This corresponds to work offset of the axis MCS. The axis is not traversed.
A preprocessing stop with synchronization is triggered by \texttt{PRESETON}. The actual position is not assigned until the axis is at standstill.

If the axis is not assigned to the channel at \texttt{PRESETON}, the further procedure depends on the axis-specific configuration of the axis interchange behavior:

\texttt{MD30552 $MA\_AUTO\_GET\_TYPE}

\textbf{Referencing status}

By setting a new actual value in the machine coordinate system, the referencing status of the machine axis is reset.

\texttt{DB31, \ldots DBX60.4/.5 = 0} (referenced/synchronized measuring system 1/2)

It is recommended that \texttt{PRESETON} only be used for axes that do not require a reference point.

To restore the original machine coordinate system, the measuring system of the machine axis must be referenced again, e.g. through active referencing from the part program (\texttt{G74}).

\begin{tabular}{|m{10cm}|}
\hline
\textbf{CAUTION} \\
\hline
\textbf{Loss of the referencing status} \\
By setting a new actual value in the machine coordinate system with \texttt{PRESETON}, the referencing status of the machine axis is reset to "not referenced/synchronized". \\
\hline
\end{tabular}

\section*{Programming}

\textbf{Syntax}

\texttt{PRESETON(<axis_1>, <value_1> [, <axis_2>, <value_2>, \ldots <axis_8>, <value_8>])}

\textbf{Meaning}

\begin{itemize}
  \item \texttt{PRESETON:} Actual value setting with loss of the referencing status
  \item Preprocessing: Yes
  \item Alone in the block: Yes
  \item \texttt{<axis_x>:} Machine axis name
  \item Type: AXIS
  \item Range of values: Machine axis names defined in the channel
  \item \texttt{<value_x>:} New actual value of the machine axis in the machine coordinate system (MCS)
  \item The input is made in the current valid system of units (inch/metric)
  \item An active diameter programming (\texttt{DIAMON}) is taken into account
  \item Type: REAL
\end{itemize}
System variable

$AC_PRESET
The axis-specific system variable $AC_PRESET provides the vector from the zero point of the currently offset MCS' to the zero point of the original MCS₀ after the referencing of the machine axis.

$AC_PRESET<axis> = $AC_PRESET<axis> + "current actual position of the axis in the MCS" - "PRESETON actual position"

The work offset can be undone with the system variables:

PRESETON(<axis>, $VA_IM + $AC_PRESET[<axis>]) ; "current actual position of the axis in the MCS'" + "offsets"

Example

Program code
N10 G1 X10 F5000
N20 PRESETON(X, $AA_IM[X]+70) ; Actual value = 10 + 70 = 80 => $AC_PRESET = $AC_PRESET - 70

Supplementary conditions

Axes for which PRESETON must not be used
- Traversing positioning axes
- Traversing command axes in spindle mode
- Traversing concurrent positioning axes (FC18)
- Axes involved in a transformation
- Traversing path axes
- Reciprocating axes
- Axes on which one or more of the following safety functions (Safety Integrated) are active
  - Enable safe limit switch
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<safe axis>], bit 1 = 1
  - Enable safe cam, pair 1 ... 4, cam +/-
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<safe axis>], bits 8 ... 15 = 1
- Hirth axes
- Synchronized axes of a gantry grouping
- Axes for which the reference point approach from the part program (G74) is active
- Slave axis of a speed/torque coupling (master-slave)
Geometry axes

- **PRESETON** can be used on a stationary geometry axis when a further geometry axis is not being traversed in the channel at the same time.

- **PRESETON** can be used on a stationary geometry axis even when a further geometry axis is being traversed in the channel at the same time, but this axis is in the "neutral axis" state or traversing as a command axis.

Example: A further geometry (X) is traversing at the same time in the "neutral axis" state

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G0 X0 Y0</td>
<td>; X, Y: Geometry axes</td>
</tr>
<tr>
<td>N15 RELEASE(Y)</td>
<td>; Neutral axis</td>
</tr>
<tr>
<td>N20 PRESETON(Y,20)</td>
<td>; Actual position Y in the MCS = 20</td>
</tr>
<tr>
<td>N30 G0 X40</td>
<td>; Geometry axis X traverses</td>
</tr>
<tr>
<td>N40 M30</td>
<td></td>
</tr>
</tbody>
</table>

Example: A further geometry (X) is traversing at the same time as command axis

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G0 X0 Y0</td>
<td>; X, Y: Geometry axes</td>
</tr>
<tr>
<td>N20 POS[X]=40 FA[X]=1000</td>
<td>; X command axis</td>
</tr>
<tr>
<td>N30 DO PRESETON(Y,20)</td>
<td>; Actual position Y in the MCS = 20</td>
</tr>
<tr>
<td>N40 M30</td>
<td></td>
</tr>
</tbody>
</table>

PLC-controlled axes

- **PRESETON** can be used on PLC-controlled axes according to their current type.

Spindle states

The following table shows the reactions that occur when **PRESETON** is used on a spindle in a synchronized action:

<table>
<thead>
<tr>
<th><strong>PRESETON in the NC program</strong></th>
<th>Traversing status</th>
<th>Assigned to the NC program</th>
<th>Main axis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Spindle mode</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed control mode</td>
<td>In motion</td>
<td>Alarm 22324</td>
<td>Alarm 22324</td>
</tr>
<tr>
<td></td>
<td>Stationary</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Positioning mode <strong>SPOS</strong></td>
<td>In motion</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Stationary</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Positioning across block bounda-</td>
<td>In motion</td>
<td>Alarm 10610</td>
<td>-</td>
</tr>
<tr>
<td>ries <strong>SPOSA</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Axis mode</td>
<td>In motion</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Stationary</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

+: Possible

-: Not possible
Axis couplings

- Leading axes: The sudden change of the leading axis position caused by PRESETON is not traversed in the following axes. The coupling is not changed.
- Following axes: Only the overlaid position component of the following axis is affected by PRESETON.

Gantry grouping

- If PRESETON is used on the guide axis of a gantry grouping, the work offset is also performed in all synchronized axes of the gantry grouping.

Indexing axes

- PRESETON can be used on indexing axes.

Software limit switches, operating range limit, protection areas

- If the axis position is outside the specified limits after a work offset by PRESETON, an alarm is not displayed until an attempt is made to traverse the axis.

Block search with calculation

PRESETON commands are collected during the block search and executed with the NC start to continue the NC program.

Position-dependent NC/PLC interface signals

- The status of the position-dependent NC/PLC interface signals is redetermined based on the new actual position.
  - Parameterized fixed point positions: MD30600 $MA_FIX_POINT_POS[0...3] = <fixed point position 1...4>
  - NC/PLC interface signals DB31, ... DBX75.3 ... 5 (JOG approach fixed point reached)
  
If the axis is at a fixed point position with the exact stop tolerance, the associated NC/PLC interface signal is set. The NC/PLC interface signal is reset when the actual value is set by PRESETON to a different value outside the exact stop tolerance around the fixed point position.

DRF offset

- A DRF offset of the axis is deleted by PRESETON.

Overlaid movement $AA_OFF

- An overlaid movement from a synchronized action with $AA_OFF is not affected by PRESETON.

Online tool offset FTOC

- An active online tool offset from a synchronized action with FTOC remains active even after PRESETON.

Axis-specific compensations

Axis-specific compensations remain active after PRESETON.
10.4.2.2 Actual value setting without loss of the referencing status (PRESETONS)

Function

The PRESETONS() procedure sets a new actual value for one or more axes in the machine coordinate system (MCS). This corresponds to work offset of the axis MCS. The axis is not traversed.

A preprocessing stop with synchronization is triggered by PRESETONS. The actual position is not assigned until the axis is at standstill.

If the axis is not assigned to the channel at PRESETONS, the further procedure depends on the axis-specific configuration of the axis interchange behavior:

MD30552 $MA_AUTO_GET_TYPE

Referencing status

By setting a new actual value in the machine coordinate system (MCS) with PRESETONS, the referencing status of the machine axis is not changed.

Requirements

- Encoder type
  PRESETONS is only possible with the following encoder types of the active measuring system:
    - MD30240 $MA_ENC_TYPE[<measuring system>] = 0 (simulated encoder)
    - MD30240 $MA_ENC_TYPE[<measuring system>] = 1 (raw signal encoder)

- Referencing mode
  PRESETONS is only possible with the following referencing mode of the active measuring system:
    - MD34200 $MA_ENC_REFP_MODE[<measuring system>] = 0 (no reference point approach possible)
    - MD34200 $MA_ENC_REFP_MODE[<measuring system>] = 1 (referencing of incremental, rotary or linear measuring systems: zero pulse on the encoder track)

Commissioning

Axis-specific machine data

Actual value setting without loss of the referencing status (PRESETONS) must be set axis-specifically:
**MD30455 $MA_MISC_FUNCTION_MASK, bit 9 = 1**

---

**Note**

**PRESETON deactivated**

Activation of the "Actual value setting without loss of the referencing status PRESETONS" function deactivates the "Actual value setting with loss of the referencing status PRESETON" function. The options mutually exclude each other.

---

**Programming**

**Syntax**

PRESETONS(<axis_1>, <value_1> [, <axis_2>, <value_2>, ... <axis_8>, <value_8>])

**Meaning**

PRESETONS: Actual value setting with loss of the referencing status
Preprocessing Yes
stop:
Alone in the block: Yes
<axis_x>: Machine axis name
Type: AXIS
Range of values: Machine axis names defined in the channel
<value_x>: New current actual value of the machine axis in the machine coordinate system (MCS)
The input is made in the active system of units (inch/metric)
An active diameter programming (DIAMON) is taken into account
Type: REAL

**System variable**

$AC_PRESET

The axis-specific system variable $AC_PRESET provides the vector from the zero point of the currently offset MCS' to the zero point of the original MCS₀ after the referencing of the machine axis.

$AC_PRESET<axis> = $AC_PRESET<axis> + "current actual position of the axis in the MCS" - "PRESETONS actual position"

The work offset can be undone with the system variables:

PRESETONS(<axis>, $VA_IM + $AC_PRESET<axis>) ; "current actual position of the axis in the MCS" + "offsets"

**Example**

Work offset of the X axis MCS by 70 units.
The programmed end position of the X axis (command axis) is transformed to the new MCS with PRESETONS.

**Program code**

```
N10 G1 X10 F5000
N20 PRESETONS(X, $AA_IM[X]+70) ; Actual value = 10 + 70 = 80 => $AC_PRESET = $AC_PRESET - 70
```

**Supplementary conditions**

**Axes for which PRESETONS must not be used**

- Traversing positioning axes
- Traversing command axes in spindle mode
- Traversing concurrent positioning axes (FC18)
- Axes involved in a transformation
- Traversing path axes
- Reciprocating axes
- Axes on which one or more of the following safety functions (Safety Integrated) are active
  - Enable safe limit switch
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<safe axis>], bit 1 = 1
  - Enable safe cam, pair 1 ... 4, cam +/-
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<safe axis>], bits 8 ... 15 = 1
- Hirth axes
- Synchronized axes of a gantry grouping
- Axes for which the reference point approach from the part program (G74) is active
- Slave axis of a speed/torque coupling (master-slave)
Geometry axes

- \texttt{PRESETONS} can be used on a stationary geometry axis when a further geometry axis is not being traversed in the channel at the same time.

- \texttt{PRESETONS} can be used on a stationary geometry axis even when a further geometry axis is being traversed in the channel at the same time, but this axis is in the "neutral axis" state or traversing as a command axis.

Example: A further geometry (X) is traversing at the same time in the "neutral axis" state

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G0 X0 Y0</td>
<td>; X, Y: Geometry axes</td>
</tr>
<tr>
<td>N15 RELEASE(Y)</td>
<td>; Neutral axis</td>
</tr>
<tr>
<td>N20 PRESETONS(Y,20)</td>
<td>; Actual position Y in the MCS = 20</td>
</tr>
<tr>
<td>N30 G0 X40</td>
<td>; Geometry axis X traverses</td>
</tr>
<tr>
<td>N40 M30</td>
<td></td>
</tr>
</tbody>
</table>

1) Note

The release of an axis in the action part of a synchronized action does not ensure that the release is on time.

Example: A further geometry (X) is traversing at the same time as command axis

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G0 X0 Y0</td>
<td>; X, Y: Geometry axes</td>
</tr>
<tr>
<td>N20 POS[X]=40 FA[X]=1000</td>
<td>; X command axis</td>
</tr>
<tr>
<td>N30 PRESETONS(Y,20)</td>
<td>; Actual position Y in the MCS = 20</td>
</tr>
<tr>
<td>N40 M30</td>
<td></td>
</tr>
</tbody>
</table>

PLC-controlled axes

- \texttt{PRESETONS} can be used on PLC-controlled axes according to their current type.

Spindle states

The following table shows the reactions that occur when \texttt{PRESETONS} is used on a spindle in a synchronized action:

<table>
<thead>
<tr>
<th>\texttt{PRESETONS} in the NC program</th>
<th>Traversing status</th>
<th>Assigned to the NC program</th>
<th>Main axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed control mode</td>
<td>In motion</td>
<td>Alarm 22324</td>
<td>Alarm 22324</td>
</tr>
<tr>
<td></td>
<td>Stationary</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Positioning mode SPOS</td>
<td>In motion</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Stationary</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Positioning across block boundaries</td>
<td>In motion</td>
<td>Alarm 10610</td>
<td>-</td>
</tr>
</tbody>
</table>

K2: Axis Types, Coordinate Systems, Frames

10.4 Coordinate systems
PRESETONS in the NC program

<table>
<thead>
<tr>
<th>Spindle mode</th>
<th>Traversing status</th>
<th>Assigned to the NC program</th>
<th>Main axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis mode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In motion</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Stationary</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

+: Possible  
-: Not possible

Axis couplings
- Leading axes: The sudden change of the leading axis position caused by PRESETONS is not traversed in the following axes. The coupling is not changed.
- Following axes: Only the overlaid position component of the following axis is affected by PRESETONS.

Gantry grouping
- If PRESETONS is used on the guide axis of a gantry grouping, the work offset is also performed in all synchronized axes of the gantry grouping.

Indexing axes
- PRESETONS can be used on indexing axes.

Software limit switches, operating range limit, protection areas
- If the axis position is outside the specified limits after a work offset by PRESETONS, an alarm is not displayed until an attempt is made to traverse the axis.

Block search with calculation
PRESETONS commands are collected during the block search and executed with the NC start to continue the NC program.

Position-dependent NC/PLC interface signals
- The status of the position-dependent NC/PLC interface signals is redetermined based on the new actual position.
  Example: Fixed point positions
  - Parameterized fixed point positions: MD30600 $MA_FIX_POINT_POS[0...3] = <fixed point position 1...4>
  - NC/PLC interface signals DB31, ... DBX75.3 ... 5 (JOG approach fixed point reached)
If the axis is at a fixed point position with the exact stop tolerance, the associated NC/PLC interface signal is set. The NC/PLC interface signal is reset when the actual value is set by PRESETONS to a different value outside the exact stop tolerance around the fixed point position.

DRF offset
- A DRF offset of the axis is deleted by PRESETONS.
Overlaid movement $AA_OFF

- An overlaid movement from a synchronized action with $AA_OFF is not affected by PRESETONS.

Online tool offset FTOC

- An active online tool offset from a synchronized action with FTOC remains active even after PRESETONS.

Axis-specific compensations

Axis-specific compensations remain active after PRESETONS.

JOG mode

- PRESETONS must only be used on a stationary axis.

JOG mode, REF machine function

- PRESETONS must not be used.

10.4.3 Basic coordinate system (BCS)

Basic coordinate system (BCS)

The basic coordinate system (BCS) consists of three mutually perpendicular axes (geometry axes) as well as other special axes, which are not interrelated geometrically.

Machine tools without kinematic transformation

BCS and MCS always coincide when the BCS can be mapped onto the MCS without kinematic transformation (e.g. TRANSMIT / face transformation, 5-axis transformation and up to three machine axes).

On such machines, machine axes and geometry axes can have the same names.

![Figure 10-14 MCS=BCS without kinematic transformation](image-url)
Machine tools with kinematic transformation

The BCS and MCS do not coincide when the BCS is mapped onto the MCS with kinematic transformation (e.g. TRANSMIT / face transformation, 5-axis transformation or more than three axes).

On such machines the machine axes and geometry axes must have different names.

![Kinematic transformation between the MCS and BCS](image)

Figure 10-15 Kinematic transformation between the MCS and BCS

Machine kinematics

The workpiece is always programmed in a two- or three-dimensional, right-angled coordinate system (WCS). However, such workpieces are being programmed ever more frequently on machine tools with rotary axes or linear axes not perpendicular to one another. Kinematic transformation is used to represent coordinates programmed in the workpiece coordinate system (rectangular) in real machine movements.

References:
Function Manual, Special Functions; 3- to 5-Axis Transformation (F2)
Function Manual, Extended Functions; Kinematic Transformation (M1)
10.4.4 Additive offsets

External work offsets

The external work offset is a linear offset between the basic coordinate system (BCS) and the basic zero system (BZS).

The external work offset $AA_{ETRANS}$ is effective in two ways depending on the machine data parameterization:

1. The $AA_{ETRANS}$ system variable has a direct effect as offset value after activation by the NC/PLC interface signal.

2. The value of the $AA_{ETRANS}$ system variable is taken into the system frame $P:EXTFRAME$ and the data management frame $P\_EXTFR$ after activation by the NC/PLC interface signal. The active complete frame $P\_ACTFRAME$ is then recalculated.

Machine data

In conjunction with the $AA_{ETRANS}$ system variable, a distinction is made between two procedures which are selected via the following machine data:

MD28082 $MC\_MM\_SYSTEM\_FRAME\_MASK$, bit1 = <value>

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Function: Direct writing of $AA_{ETRANS}[&lt;axis&gt;]$ by the PLC, HMI or NC program. Enable to traverse the work offset of $AA_{ETRANS}[&lt;axis&gt;]$ in the next possible traversing block: DB31, ... DBX3.0</td>
</tr>
</tbody>
</table>
| 1      | Function: Activation of the active system frame $P\_EXTFRAME$ and the data management frame $P\_EXTFR$. Enable to traverse the work offset of $AA_{ETRANS}[<axis>]$ through: DB31, ... DBX3.0. Then in the channel:  
- Stop of all traversing movements in the channel (except command and PLC axes)  
- Preprocessing stop with subsequent reorganization (STOPRE)  
- Coarse offset of the active frame $P\_EXTFRAME[<axis>] = AA_{ETRANS}[<axis>]$  
- Coarse offset of the data management frame $P\_EXTFR[<axis>] = AA_{ETRANS}[<axis>]$  
- Recalculation of the active complete frame $P\_ACTFRAME$  
- Traversing of the offset in the programmed axes.  
- Continuation of the interrupted traversing movement or the NC program |
Programming

- Syntax
  \[\$AA_{\text{ETRANS}}[<\text{axis}>] = <\text{value}>\]

- Meaning

<table>
<thead>
<tr>
<th>$AA_{\text{ETRANS}}$:</th>
<th>System variable to buffer the external work offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;axis&gt;:</td>
<td>Channel axis</td>
</tr>
<tr>
<td>&lt;value&gt;:</td>
<td>Offset value</td>
</tr>
</tbody>
</table>

NC/PLC interface signal

DB31, ... DBX3.0 = 0 \(\rightarrow\) $P_{\text{EXTFRAME}}[<\text{axis}>] = $P_{\text{EXTFR}}[<\text{axis}>] = $AA_{\text{ETRANS}}[<\text{axis}>]

DRF offset

The DRF offset enables the adjustment of an additive incremental work offset for geometry and additional axes in the basic coordinate system through handwheel.

The DRF offset can be read via the axis-specific system variable:

$AC_{\text{DRF}}[<\text{Axis}>]

References:
Function Manual, Extended Functions; Manual and Handwheel Travel (H1), Section: DRF offset

Overlaid movements

The "Overlaid movement" for the programmed axis can only be accessed from synchronized actions via the system variable $AA_{\text{OFF}}[\text{axis}].

RESET response

Channel reset / end of program reset
The activated values remain active after RESET and program end.
Reset response of channel-specific system frames as follows:

- The system frame for the "external work offset" is active after RESET with the following machine data setting:
  MD24006 $MC_{\text{CHSFRAME_RESET\_MASK}}, \text{bit } 1 = 1

- The "external work offset" in the active system frame is deleted in the data management through the following machine data setting:
  MD24006 $MC_{\text{CHSFRAME_RESET\_MASK}}, \text{bit } 1 = 0

The following frames are active after RESET:

- System frame for:
  MD24006 $MC_{\text{CHSFRAME_RESET\_MASK}}, \text{Bit } 4 = 1 (\text{workpiece reference point})
  MD24006 $MC_{\text{CHSFRAME_RESET\_MASK}}, \text{Bit } 5 = 1 (\text{cycles})
Power On
After control run-up (Power On), the last used offset values of the external work offset are stored and do not become effective again until there is a renewed activation signal.

System frames are retained during Power ON, depending on the following machine data:
MD24008 $MC_CHSFRAME_POWERON_MASK (reset system frames after Power On)

Suppression: External work offset
- The SUPA command suppresses the external work offset while the block is being processed.
- Active external work offsets are suppressed for the duration of the reference point approach through the G74 command (reference point approach) and the equivalent operator actions in the reference point approach mode.
- With G74, i.e. "Automatic" or "MDA" mode, the previously active external work offset automatically becomes active again with the next traversing movement in the block.
- After a mode change from the reference point approach mode, the NC/PLC interface signal for the referenced axes must be set for reactivation.

10.4.5 Basic zero system (BZS)

Basic zero system (BZS)
The basic zero system (BZS) is the basic coordinate system with a basic offset.

![Diagram of Basic offset between BCS and BZS](image)

Figure 10-16 Basic offset between BCS and BZS
**Basic offset**

The basic offset describes the coordinate transformation between BCS and BZS. It can be used, for example, to define the palette window zero.

The basic offset comprises:

- Zero offset external
- DRF offset
- Superimposed motion
- Chained system frames
- Chained basic frames

![Diagram of coordinate systems and offset relationships](image)

**Figure 10-17** Example of the use of the basic offset

The following settings apply:

- The user can change the basic offset from the part program by means of an operator action and from the PLC.
- If the basic offset is to take effect immediately, an ASUB can be started via the PLC using FC9 in order to execute the appropriate G code.

**Note**

**Recommendation to the machine manufacturer**

Use the 3rd basic offset onwards for your own applications.

The 1st and 2nd basic offset are reserved for PRESET and the "Zero offset external".
10.4.6 Settable zero system (SZS)

Settable zero system (SZS)

The "settable zero system" (SZS) is the workpiece coordinate system WCS with a programmable frame (viewed from the perspective of the WCS). The workpiece zero is defined by the settable FRAMES G54 to G599.

Programmable offsets act on the "settable zero system". All programmable offsets refer to the "settable zero system".

WCS actual-value display in WCS or SZS

The actual values of the axes in the machine coordinate system (MCS) or the WCS can be displayed on the HMI operator interface. For displays in WCS, the actual values can also be displayed in relation to the SZS. The corresponding parameterization takes place through the machine data:

MD9424 $MM_MA_COORDINATE_SYSTEM (coordinate system for actual value display)

<table>
<thead>
<tr>
<th>Value</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Actual-value display in relation to the WCS</td>
</tr>
<tr>
<td>1</td>
<td>Actual-value display in relation to the SZS</td>
</tr>
</tbody>
</table>
Note
Display of the current coordinate system

When "Actual-value display in relation to the SZS" is active, the WCS is still displayed on the HMI operator interface as the coordinate system to which the actual-value display relates.

Example

Actual-value display in relation to the WCS or SZS

<table>
<thead>
<tr>
<th>Code (excerpt)</th>
<th>Actual value display: Axis X (WCS)</th>
<th>Actual value display: Axis X (SZS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 X100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>N20 X0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>N30 $P_PFRAME = CTRANS(X,10)</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>N40 X100</td>
<td>100</td>
<td>110</td>
</tr>
</tbody>
</table>

10.4.7 Workpiece coordinate system (WCS)

Workpiece coordinate system (WCS)

The workpiece coordinate system (WCS) is the programming basis.
10.5 Frames

10.5.1 Frame types

A frame is a data structure that contains values for offset (TRANS), fine offset (FINE), rotation (ROT), mirroring (MIRROR) and scaling (SCALE) for axes.

When activating the frame, using the frame values, a static coordinate transformation for the axes contained in the frame is performed using a defined algorithm.

Axial frame

An axial frame contains the frame values of an axis.

Example of the data structure of an axial frame for axis X:

<table>
<thead>
<tr>
<th>Axis</th>
<th>TRANS</th>
<th>FINE</th>
<th>ROT</th>
<th>MIRROR</th>
<th>SCALE</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>10.0</td>
<td>0.1</td>
<td>0.0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Channel-specific frame

A channel-specific frame contains frame values for all channel axes (geometry, special and machine axes).

Rotations (ROT) are only included in the calculation for geometry axes.

A channel-specific frame is only active in the channel in which the frame is defined.

Example of the data structure of a channel-specific frame:

- Geometry axes: X, Y, Z
- Special axes: A
- Machine axes: AX1

<table>
<thead>
<tr>
<th>Axis</th>
<th>TRANS</th>
<th>FINE</th>
<th>ROT</th>
<th>MIRROR</th>
<th>SCALE</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>10.0</td>
<td>0.1</td>
<td>0.0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Y</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Z</td>
<td>0.0</td>
<td>0.0</td>
<td>45.0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>2.0</td>
<td>0.1</td>
<td>0.0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>AX1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Global frame

A global frame contains the frame values for all machine axes.

A global frame is active in all channels of the NC.

Example of the data structure of a channel-specific frame:

- Machine axes: AX1, ... AX5
10.5.2 Frame components

10.5.2.1 Translation

Programming

The programming of the translation or coarse offset can be performed via the following commands:

- **Example of data management frames** $P_{UIFR}$
  
  - **Complete frame**: $P_{UIFR}[<n>] = CTRANS(<K1>,<V1>[,<K2>,<V2>]
    [,<K3>,<V3>])$
    with Km = coordinate x, y or z and Vm = offset m
  
  - **Frame component**: $P_{UIFR}[<n>,<k>,TR] = <V>$
    with K = coordinate x, y or z and V = offset

- **Example of programmable frame**
  
  - **TRANS** <K1> <V1> [<K2> <V2>] [<K3> <V3>]
    with Km = coordinate x, y or z and Vm = offset m

Program examples:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{UIFR}[1] = CTRANS(X,10,Y,10)</td>
<td>Complete frame</td>
</tr>
<tr>
<td>$P_{UIFR}[1,X,TR] = 10</td>
<td>Frame components</td>
</tr>
<tr>
<td>TRANS X=10 Y=10</td>
<td>Programmable frame</td>
</tr>
</tbody>
</table>
10.5 Frames

10.5.2.2 Fine offset

Parameterization

The fine offset is enabled via the machine data:

MD18600 $MN_MM_FRAME_FINE_TRANS = <value>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The fine offset cannot be entered or programmed.</td>
</tr>
<tr>
<td>1</td>
<td>Fine offset is possible for settable frames, basic frames and the programmable frame via command or program.</td>
</tr>
</tbody>
</table>

Programming

The programming of the translation or coarse offset can be performed via the following commands:

- **Example of data management frames** $P_UIFR
  - Complete frame: $P_UIFR[n] = CFINE(K1,V1,K2,V2,K3,V3)
    - with Km = coordinate x, y or z and Vm = offset m
  - Frame component: $P_UIFR[n,K,F1] = V
    - with K = coordinate x, y or z and V = offset

- **Example of programmable frame**
  - TRANS K1 V1 [K2 V2][K3 V3]
    - with Km = coordinate x, y or z and Vm = offset m

Programming examples:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_UIFR[1] = CTRANS(X,10,Y,10)</td>
<td>Complete frame</td>
</tr>
</tbody>
</table>
10.5.2.3 Rotation Overview (geometry axes only)

Function

The direction of rotation around the coordinate axes is determined by means of a right-hand, rectangular coordinate system with axes x, y and z. If the rotary motion is in a clockwise direction when looking in the positive direction of the coordinate axis, the direction of rotation is positive. A, B and C designate rotations whose axes are parallel to the coordinate axes.

The following figure shows the new position of the coordinate system x', y' and z' after the rotation around z with γ = -45°.

Parameterization of the rotation sequence

The following machine data is used to set around which coordinate axes and in which order the rotations are performed when more than one angle of rotation is programmed:
MD10600 \$MN\_FRAME\_ANGLE\_INPUT\_MODE = <value>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Euler angles in z'y'' convention (RPY angles)</td>
</tr>
<tr>
<td>2</td>
<td>Euler angles in z'x'' convention</td>
</tr>
</tbody>
</table>

**Note**

For historical reasons, Euler angles in z'x'' convention can be used. However, it is strongly recommended that only Euler angles in z'y'' convention (RPY angles) be used (see Section Rotation with a Euler angles: ZY'X'' convention (RPY angles) (Page 708)).

### 10.5.2.4 Rotation with a Euler angles: ZY'X'' convention (RPY angles)

Euler angles in z'y'' convention are also called RPY angles. RPY is derived from:

- R: Roll → rotation around x
- P: Pitch → rotation around y'
- Y: Yaw → rotation around z''

With RPY angles, the rotations are in the order z, y', x''.

#### Assignment of rotary axis to geometry axis

<table>
<thead>
<tr>
<th>Rotary axis</th>
<th>Geometry axis in channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>x''</td>
<td>1st geometry axis</td>
</tr>
<tr>
<td>y'</td>
<td>2nd geometry axis</td>
</tr>
<tr>
<td>z</td>
<td>3rd geometry axis</td>
</tr>
</tbody>
</table>
Value range
With RPY angles, programmed values can only be unambiguously calculated back within the following value ranges:

\[-180 \leq x \leq 180\]
\[-90 < y < 90\]
\[-180 \leq z \leq 180\]

Programming of the complete frame
When programming the complete frame, all rotation components of the frame are always written. Non-programmed components are implicitly assigned the value 0°.

Syntax

\[
\text{<Frame>} = \text{CROT}([\text{<1st GAx>},\text{<angle>},][\text{<2nd GAx>},\text{<angle>},][\text{<3rd GAx>},\text{<angle>}])
\]
\[
\text{ROT} \ [\text{<1st GAx>},\text{<angle>}] \ [\text{<2nd GAx>},\text{<angle>}] \ [\text{<3rd GAx>},\text{<angle>}]
\]
\[
\text{AROT} \ [\text{<1st GAx>},\text{<angle>}] \ [\text{<2nd GAx>},\text{<angle>}] \ [\text{<3rd GAx>},\text{<angle>}]
\]

Meaning

\text{CROT}: \quad \text{Absolute rotation, reference frame: Arbitrary programmed frame}
\text{<Frame>}: \quad \text{Arbitrary active or data management frame}
\text{ROT}: \quad \text{Absolute rotation, reference frame: Programmable frame $P\text{PFRAME}$, reference point: Zero point of the current workpiece coordinate system set with G54 ... G57, G505 ... G599}
\text{AROT}: \quad \text{Additive rotation, reference frame: Programmable frame $P\text{PFRAME}$, reference point: Zero point of the current workpiece coordinate system set with G54 ... G57, G505 ... G599}
\text{<nth GAx>}: \quad \text{Name of the nth geometry axis around which rotation is to be performed with the specified angle. The value 0° is implicitly set as angle of rotation for a geometry axis that has not been programmed. Assignment of geometry axis to rotary axis:}
\begin{align*}
\text{Geometry axis} & \quad \text{Rotary axis} \\
1\text{st geometry axis} & \quad x'' \\
2\text{nd geometry axis} & \quad y' \\
3\text{rd geometry axis} & \quad z
\end{align*}
\text{<angle>}: \quad \text{Angle specification in degrees.}

Programming a frame component
When programming a frame component, only the programmed component of the frame is written. The components that are not programmed remain unchanged.

Syntax

\[
\text{<frame>}[\text{<index>,<GAx>},\text{RT}] = \text{<angle>}
\]
Meaning

<Frame>: Arbitrary active or data management frame
ндекс: Array index of the frame, e.g. $P\text{_{UIFR}}[0 ... n]
<GAxis>: Name of the geometry axis around which rotation is to be performed with the specified angle.
<Angle>: Angle specification in degrees.

Reading back the rotation components

In general, the same values are obtained when reading back the rotation components of a frame as those that were programmed:

<table>
<thead>
<tr>
<th>Programmed</th>
<th>Saved rotation components ¹)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x, RT</td>
</tr>
<tr>
<td>(&lt;\text{Frame}&gt;=\text{CROT}(X,45,Y,30,Z,-20))</td>
<td>45</td>
</tr>
</tbody>
</table>

¹) The values of the saved rotation components are obtained when reading back

Values outside the value range

Programmed values outside a value range are mapped on the range limits:

<table>
<thead>
<tr>
<th>Programmed</th>
<th>Saved rotation components ¹)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x, RT</td>
</tr>
<tr>
<td>(&lt;\text{Frame}&gt;=\text{CROT}(X,190,Y,0,Z,-200))</td>
<td>-170</td>
</tr>
</tbody>
</table>

¹) The values of the saved rotation components are obtained when reading back

Note

It is recommended that when writing the rotation components of the frame, the specified value ranges are observed so that the same values are obtained when reading back the rotation components.

Gimbal lock

Gimbal lock designates a geometric problem in which the rotation components can no longer be unambiguously calculated back from the position vector. Gimbal lock occurs in RPY angles with an angular position of the rotation component $y = 90°$. In this case, the rotation components are converted by the controller after being written so that the following applies:

- Rotation component $z = \text{rotation component } z - \text{rotation component } x$
- Rotation component $x = 0°$
- Rotation component $y = 90°$
Programmed | Saved rotation components
--- | ---
\(<\text{Frame}\>=\text{CROT}(X,30,Y,90,Z,40)\) | \(x, \text{RT}\) \(y, \text{RT}\) \(z, \text{RT}\)
| 0 | 90 | 40 - 30 = 10

⚠️ CAUTION

Different values for reading back the rotation component \(z\)

Because of the different conversion times after writing the complete frame or the writing of individual rotation components of a data management frame and the writing of individual rotation components of an active frame, different values can be read back for rotation component \(z\).

Differences when writing the complete frame and frame components

Two cases must be distinguished when writing the rotation components of a frame:

1. Writing the complete frame: \(<\text{Frame}\>=\text{CROT}(X,a,Y,b,Z,c)\)
   When writing the complete frame, the conversion is immediately at the time of writing.

2. Writing individual rotation components, e.g. rotation around \(X\): \(<\text{Frame}\>[0,X,\text{RT}]\)=a
   When writing individual rotation components, the conversion depends on the storage location of the frame:
   - Data management frames
     With data management frames, the conversion is at the time of activation of the frame based on the rotation components written by this time. With regard to the conversion of a data management frame, a data management frame therefore behaves in the same way after writing individual rotation components as when writing the complete frame.
   - Active frames
     In the case of active frames, the conversion is immediately at time of writing of the rotation component.

Examples: Writing the complete frame

The conversion is made in each block after the complete frame has been written.

<table>
<thead>
<tr>
<th>Programmed</th>
<th>Saved rotation components</th>
</tr>
</thead>
</table>
| \(<\text{Frame}\>=\text{CROT}(X,0,Y,90,Z,90)\) | \(x, \text{RT}\) \(y, \text{RT}\) \(z, \text{RT}\)
| N10 | 0 | 90 | 90
| N20 | 0 | 90 | -90 \(^1\)
| N30 | 0 | 90 | 0 \(^1\)

1) Different values compared to the writing of individual rotation components of an active frame
Examples: Writing individual rotation components of a data management frame

A conversion is performed on the activation of the data management frame. In the example, at any time after N30.

<table>
<thead>
<tr>
<th>Programmed</th>
<th>Saved rotation components</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x, RT</td>
</tr>
<tr>
<td>N10 &lt;data management frame&gt;</td>
<td>0</td>
</tr>
<tr>
<td>N20 &lt;data management frame&gt;</td>
<td>0</td>
</tr>
<tr>
<td>N30 &lt;data management frame&gt;</td>
<td>0</td>
</tr>
</tbody>
</table>

1) Different values compared to the writing of individual rotation components of an active frame.

Examples: Write individual rotation components of an active frame

Any required conversion is performed immediately on writing the rotation component. The stored initial values of the frame are: \(x = 0, y = 0, z = 0\).

<table>
<thead>
<tr>
<th>Programmed</th>
<th>Saved rotation components</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x, RT</td>
</tr>
<tr>
<td>N10 &lt;active frame&gt;</td>
<td>0</td>
</tr>
<tr>
<td>N20 &lt;active frame&gt;</td>
<td>0</td>
</tr>
<tr>
<td>N30 &lt;active frame&gt;</td>
<td>0</td>
</tr>
</tbody>
</table>

1) Different values compared to the writing of the complete frame or the writing of individual rotation components of a data management frame.
10.5.2.5 Rotation with a Euler angles: ZX'Z'' convention

With Euler angles, the rotations are in the order z, x', z''.

**Note**

**Recommended use**

For historical reasons, Euler angles in zx'z'' convention can be used. However, it is strongly recommended that only Euler angles in zy'x'' convention (RPY angles) be used (see Section Rotation with a Euler angles: ZY'X'' convention (RPY angles) (Page 708)).

**Assignment of rotary axis to geometry axis**

<table>
<thead>
<tr>
<th>Rotary axis</th>
<th>Geometry axis in channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>z''</td>
<td>3rd geometry axis</td>
</tr>
<tr>
<td>x'</td>
<td>1st geometry axis</td>
</tr>
<tr>
<td>z</td>
<td>3rd geometry axis</td>
</tr>
</tbody>
</table>

**Range of values**

Data from Euler angles can only be unambiguously calculated back within the following value ranges:

\[
0 \leq x < 180 \\
-180 \leq y \leq 180 \\
-180 \leq z \leq 180
\]
For data outside the specified value ranges, a modulo conversion is made referred to the value of the particular range limit.

Note

It is recommended that when writing the rotation components of the frame, the specified value ranges are observed so that the same values are obtained when reading back the rotation components.

10.5.2.6 Rotation in any plane

CRPL - Constant Rotation Plane

The predefined function "Constant Rotation Plane" enables a rotation to be programmed for a frame in an arbitrary plane \((G17, G18, G19)\) without specifying the name of a geometry axis. This enables rotations to be programmed in the third plane when only two geometry axes are present in the channel due to the specific machine constellation.

Syntax

\[ \text{CRPL(<rotary axis>,<angle of rotation>)} \]

Meaning

<table>
<thead>
<tr>
<th>CRPL:</th>
<th>Rotation in any plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;\text{rotary axis}&gt;):</td>
<td>Axis around which the rotation is performed</td>
</tr>
<tr>
<td>Type:</td>
<td>INT</td>
</tr>
<tr>
<td>Value</td>
<td>Meaning</td>
</tr>
<tr>
<td>0</td>
<td>Rotation in the active plane</td>
</tr>
<tr>
<td>1</td>
<td>Rotation around Z</td>
</tr>
<tr>
<td>2</td>
<td>Rotation around Y</td>
</tr>
<tr>
<td>3</td>
<td>Rotation around X</td>
</tr>
</tbody>
</table>

\(<\text{angle of rotation}>\): Angle in degrees through which the rotation is performed

<table>
<thead>
<tr>
<th>Type:</th>
<th>REAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is strongly recommended to observe the specified angular ranges. If the limits are not observed, then an unambiguous reverse calculation is not possible. Angles outside the limits are not rejected.</td>
<td></td>
</tr>
<tr>
<td>RPY angle:</td>
<td>( X \quad -180 \leq &lt;\text{angle of rotation}&gt; \leq 180 )</td>
</tr>
<tr>
<td></td>
<td>( Y \quad -90 \leq &lt;\text{angle of rotation}&gt; \leq 90 )</td>
</tr>
<tr>
<td></td>
<td>( Z \quad -180 \leq &lt;\text{angle of rotation}&gt; \leq 180 )</td>
</tr>
<tr>
<td>ZX'Z&quot; convention:</td>
<td>( X \quad -180 \leq &lt;\text{angle of rotation}&gt; \leq 180 )</td>
</tr>
<tr>
<td></td>
<td>( Y \quad 0 \leq &lt;\text{angle of rotation}&gt; \leq 180 )</td>
</tr>
<tr>
<td></td>
<td>( Z \quad -180 \leq &lt;\text{angle of rotation}&gt; \leq 180 )</td>
</tr>
</tbody>
</table>
Chaining with frames

CRPL() can be chained with frames and known frame functions such as CTRANS(), CROT(), CMIRROR(), CSCALE(), CFINE() etc.

Examples:

$P\_PFRAME = P\_PFRAME : CRPL(0,30.0)

$P\_PFRAME = CTRANS(X,10) : CRPL(1,30.0)

$P\_PFRAME = CROT(X,10) : CRPL(2,30.0)

$P\_PFRAME = CRPL(3,30.0) : CMIRROR(Y)

10.5.2.7 Scaling

Programming

The program commands below are used to program the scaling:

$P\_UIFR[1] = CSCALE(x,1,y,1)

SCALE x = 1 y = 1

$P\_UIFR[1,x,sc] = 1
10.5.2.8 Mirroring

Programming

The program commands below are used to program a mirroring:

$P_{UIFR[1]} = \text{CMIRROR}(x, 1, y, 1)

MIRROR x = 1
y = 1

$P_{UIFR[1, x, mi]} = 1

10.5.2.9 Chain operator

Frame components or complete frames can be combined into a complete frame using the chain operator (:).

10.5.2.10 Programmable axis name

Geometry, channel and machine axis names can be used in the frame commands. The programmed axis must be known to the channel-specific frames in the channel.

SPI

When programming frame commands, the \text{SPI}(<\text{spindle number}>)$ axis function can be used in place of an axis name.

\text{SPI}(<\text{spindle number}>)$ forms the reference of the spindle to the channel axis.

→ refer to MD35000 \$MA_SPIND_ASSIGN_TO_MACHAX[] (assignment of spindle to machine axis)

The following frame commands can be programmed with \text{SPI}(\text{spino}):

\text{TRANS()}
\text{CFINE()}
\text{CMIRROR()}
CSCALE()

A spindle can only be assigned to one rotary axis at a time. The CROT(...) function can therefore not be programmed with SPI(), as only geometry axes are permitted for CROT().

The channel axis name or machine axis name of the axis belonging to the spindle is always output when decompiling frames, even when axis names have been programmed in the part program with SPI(...).

If the spindle is assigned e.g., to the Channel Axis A then the programming:

\[
N10 \ P_{UIFR}[1] = \text{CTRANS} (SPI(1), 33.33, X, 1):\text{CSCALE} (SPI(1), 33.33):\text{CMIRROR} (SPI(1))
\]

during recompilation:

\[
P_{UIFR}[1]=\text{CTRANS} (X, 1, A, 33.33):\text{CSCALE} (A, 33.33):\text{CMIRROR} (A)
\]

If a spindle and an assigned axis are programmed in a frame command, then Alarm 16420 "Axis % multiply programmed" is output.

**Example:**

\[
P_{UIFR}[1] = \text{CTRANS} (SPI(1), 33.33, X, 1, A, 44)
\]

(The spindle is assigned to Axis A.)

**Programming examples**

\[
\text{\$P\_PFRAME} [SPI(1), TR]=22.22
\]

\[
\text{\$P\_PFRAME} = \text{CTRANS} (X, \text{axis value}, Y, \text{axis value}, SPI(1), \text{axis value})
\]

\[
\text{\$P\_PFRAME} = \text{CSCALE} (X, \text{Scale}, Y, \text{scale}, SPI(2), \text{scale})
\]

\[
\text{\$P\_PFRAME} = \text{CMIRROR} (S1, Y, Z)
\]

\[
\text{\$P\_UBFR} = \text{CTRANS} (A, 10):\text{CFINE} (SPI(1), 0.1)
\]
10.5.2.11 Coordinate transformation

The formulae below are used to discover the coordinate transformation for geometry axes:

\[
\begin{align*}
\text{WCS} \rightarrow \text{BCS} & : \quad \vec{v} = R \cdot S \cdot M \cdot \vec{v}' + \vec{t} \\
\text{BCS} \rightarrow \text{WCS} & : \quad \vec{v}' = \text{inv}(M) \cdot \text{inv}(S) \cdot \text{inv}(R) \cdot (\vec{v} - \vec{t})
\end{align*}
\]

\(\vec{v}\): Position vector in BCS

\(\vec{v}'\): Position vector in WCS

10.5.3 Data management frames and active frames

10.5.3.1 Overview

Frame types

The following frame types are available:

- System frames ($P\_PARTFR$, ... see figure)
- Basic frames ($P\_NCBFR[<n>], P\_CHBFR[<n>]$)
- Grinding frames ($P\_GFR[<n>]$)
- Settable frames ($P\_UIFR[<n>]$)
- Programmable frame ($P\_PFRAME$)
For all frame types except the programmable frame, one or more frames exist in the data management (data management frames) in addition to the frame active in the channel. For the programmable frame, only the frame active in the channel exists.

**Writing frames**

Data management frames and active frames can be written from the part program. Only data management frames can be written via the user interface.
Archiving frames
Only data management frames can be archived.

10.5.3.2 Activating data management frames
Data management frames become active frames as a result of the following actions:

- "Settable frames" G group: G54 ... G57, G500, G505 ... G599
- "Grinding frames" G group: GFRAME0 ... GFRAME100
- RESET and MD20110 $MC_RESET_MODE_MASK, bit14 == 1 (the current setting of the basic frame is retained)
- Transformation changeover
- Changing the geometry axis assignment GEOAX
- From the HMI with PI service "_N_SETUDT"

Activation from the HMI
The activation of a data management frame from the HMI with PI service "_N_SETUDT" only becomes active in the channel after a hot restart for the current part program.

The activation is effective in the reset state if the following machine data is set:

MD9440 $MM_ACTIVATE_SEL_USER_DATA (set active offset immediately)

Activating system frames
System frames are activated by:

- Programming the corresponding system function in the part program
- Operator control at SINUMERIK Operate

Note
Modifying system frames of the data management

In principle, system frames of the data management can be modified by the cycle programmer and activated using a G500, G54...G599 command. However, this option should only be used with reservation.

Activating data management frames
The behavior when activating data management frames is set using the following machine data:
MD24050 $MC_FRAME_SAA_MODE (save and activate data management frames)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>@@@</td>
</tr>
</tbody>
</table>
| 1   | 1     | Data management frames are only activated by programming the system variables $P_CHBFRMASK, $P_NCBFRMASK and $P_CHSFRMASK.  
|     |       | G500...G599 activates the appropriate settable frame. |
| 1   | 0     | Data management frames are implicitly described by functions, such as TOROT, PAROT, external work offset and transformations. |
| 1   | 1     | Data management frames are not implicitly described by functions, such as TOROT, PAROT, external work offset and transformations. |

**Activation of system frames via system variable $P_CHSFRMASK**

The system frames of the data management can be activated using system variable $P_CHSFRMASK. The value of the variables is specified as bit coded according to the machine data:

MD28082 $MC_MM_SYSTEM_FRAME_MASK (system frames of the data management)

The corresponding system frame of the data management in the channel is activated by setting a bit of system variable $P_CHSFRMASK to 1. For a value of 0, the currently active system frame in the channel remains active.

**Activating system frames after RESET**

After RESET, the system frames in the channel are activated whose bits are set in the following machine data:

MD24006 $MC_CHSFRAME_RESET_MASK (active system frames after Reset)

**Activating system frames for TCARR, PAROT and TOROT, TOFRAME**

The system frames for TCARR, PAROT and TOROT, TOFRAME are activated according to the setting in the following machine data:

MD20150 $MC_GCODE_RESET_VALUES (initial setting of the G groups)

When changing over geometry axes using transformation selection/deselection or the GEOAX command, the actual total frame $P_ACTFRAME is either deleted or is re-calculated using the new geometry axis constellation and activated. The system frames and all other frames are conditioned again in relation to the geometry axes.

**10.5.3.3 NCU-global and channel-specific frames**

- Settable frames and grinding frames can only be configured as NCU-global or channel-specific frames.
- Basic frames can be configured as NCU-global and channel-specific frames.
- An NCU-global frame has the same effect in all channels of the NCU.
• All channels of an NCU can read and write NCU-global frames equally.

• As the assignment of machine axes to channel axes and, in particular, to geometry axes, can be different in all channels, there is consequently no unique cross-channel geometric relationship between the channel axes. Therefore, only offset, scaling and mirroring is possible for NCU-global frames. Rotations are not possible.

Note

Program coordination

The coordination of channel-specific accesses to NCU-global frames is the sole responsibility of the user. It is recommended that the commands for the program coordination be used.

References

Programming Manual, Job Planning; Section "Flexible NC programming" > "Program coordination (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM)"

10.5.4 Frame chain and coordinate systems

10.5.4.1 Overview

The figure below shows the frame chain for the current complete frame. The frame chain is located between the basic coordinate system (BCS) and the workpiece coordinate system (WCS).

The settable zero system (SZS) corresponds to the WCS transformed by the programmable frame. The basic zero system (BZS) still includes the current settable frame. The system frame for the external work offset exists only if it has been configured. Otherwise, the external work offset is traversed as overlaid movement of the axis.
Complete frame

The current complete frame $P_{ACTFRAME}$ results from the chaining of all active frames of the frame chain:

$$P_{ACTFRAME} = P_{PARTFRAME} : P_{SETFRAME} : P_{EXTFRAME} : P_{ISO1FRAME} : P_{ISO2FRAME} : P_{ISO3FRAME} : P_{ACTBFRAME} : P_{IFRAME} : P_{GFRAME} : P_{TOOLFRAME} : P_{WPFRAME} : P_{TRAFRAME} : P_{PFRAME} : P_{ISO4FRAME} : P_{CYCFRAME}$$

10.5.4.2 Relative coordinate systems

Relative coordinate systems display the current setpoint positions of the axes which lie relative to a specified reference point in the active displayed coordinate system. No programming can be done regarding the relative coordinate systems. Only the axis positions in these systems can be read via the system variables.

The new display coordinate systems lie relative to WCS and ENS coordinate system and result through transformation of the WCS or ENS axis positions with the active system frame $P_{RELFRAME}$. The relative coordinate systems can not only be displaced linearly, but also rotated, mirrored, compressed or expanded.
The position indicator for axis setpoints is done in WCS or in ENS. The configuring is done via HMI machine data. Always only one display-coordinate system is active in the channel. For this reason only one relative frame is provided which generates both relative coordinate systems in the same ratio. The HMI displays the relative coordinates according to the configuration.

![Relative coordinate systems](image)

**Figure 10-21 Relative coordinate systems**

The data maintenance frame $P_{RELFR}$ can be written in the part program and via BTSS. All the frame components can be modified.

The active system frame $P_{RELFRAME}$ can be written in the part program and via BTSS.

The configuring of the system frame $P_{RELFR}$ is done via the following machine data:

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD28082 $MC_{MM_SYSTEM_FRAME_MASK}$</td>
<td>11</td>
<td>Creation of $P_{RELFR}$; with this, relative coordinate systems become existent.</td>
</tr>
<tr>
<td>MD28083 $MC_{MM_SYSTEM_DATAFRAME_MASK}$</td>
<td>11</td>
<td>Data maintenance frame $P_{RELFR}$</td>
</tr>
<tr>
<td>MD24006 $MC_{CHSFRAME_RESET_MASK}$</td>
<td>11</td>
<td>$P_{RELFR}$ becomes active at Reset</td>
</tr>
<tr>
<td>MD24007 $MC_{CHSFRAME_RESET_CLEAR_MASK}$</td>
<td>11</td>
<td>$P_{RELFR}$ is deleted at Reset</td>
</tr>
<tr>
<td>MD24008 $MC_{CHSFRAME_POWERON_MASK}$</td>
<td>11</td>
<td>$P_{RELFR}$ is deleted at PowerOn</td>
</tr>
</tbody>
</table>

The axis position in the relative coordinate system $WCS_{Rel}$ can be read via the variable $AA_{PCS\_REL}[ax]$. The variable can be read in part program, BTSS and via synchronized actions.

The axis position in the relative coordinate system $ENS_{Rel}$ can be read via the variable $AA_{ACS\_REL}[ax]$. The variable can be read in part program, BTSS and via synchronized actions.
The setting of a relative reference point via the operator panel is done via the general command interface for the workpiece and tool measuring. The system frame \$P\_RELFR for relative coordinate systems is calculated and activated as follows:

- \$AC\_MEAS\_TYPE = 14
- PI-services _N_SETUDT(6, 7)

An example of setting the relative axis positions can be found in:

**References:**
Function Manual, Extended functions; Measurement (M5),
Section "Measurement of geometry and special axes (meas. type 14, 15)"

### 10.5.4.3 Selectable SZS

Within a cycle, machining is performed in a cycle-specific workpiece coordinate system (WCS). The cycle-specific WCS results from SZS transformed by the programmable frame \$P\_PF\_FRAME and/or cycle frame \$P\_CYC\_FRAME programmed for the cycle.

If a cycle is interrupted by a machine operator, e.g. through an NC stop, traversing should then be performed in the original coordinate system (SZS) valid before the activation of the cycle.

**Machine data**

The specification how the SZS is to be calculated from the cycle-specific WCS is set via the following machine data.

MD24030 \$MC\_FRAME\_ACS\_SET = <value>

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SZS = WCS transformed with $P_TRA_FRAME, $P_PF_FRAME, $P_ISO4_FRAME and $P_CYC_FRAME</td>
</tr>
<tr>
<td>1</td>
<td>SZS = WCS transformed only with $P_CYC_FRAME</td>
</tr>
</tbody>
</table>

1. MD24030 \$MC\_FRAME\_ACS\_SET = 0
2. MD24030 \$MC\_FRAME\_ACS\_SET = 1

**Diagram:**

1. \$P\_TRA\_FRAME : \$P\_PF\_FRAME : \$P\_ISO4\_FRAME : \$P\_CYC\_FRAME
2. \$P\_TRA\_FRAME : \$P\_PF\_FRAME : \$P\_ISO4\_FRAME : \$P\_CYC\_FRAME

**Legend:**

1. SZS = WCS transformed with \$P\_TRA\_FRAME, \$P\_PF\_FRAME, \$P\_ISO4\_FRAME and \$P\_CYC\_FRAME
2. SZS = WCS transformed only with \$P\_CYC\_FRAME
Effects

The reconfiguration of the SZS has an effect on:

- SZS-related actual values: Actual-value displays, system variables, e.g. $AA_IEN, etc.
- Manual traversing (JOG) of geometry axes in the SZS

10.5.4.4 Manual traversing of geometry axes either in the WCS or in the SZS ($AC_JOG_COORD)

Previously, geometry axes have been traversed manually in JOG mode in the WCS. In addition, there is also the option to carry out this manual operation in the SZS coordinate system. The $AC_JOG_COORD variable enables the user to switch between manual traversing in the WCS and SZS. The user can now select if he wants to traverse in the SZS or the WCS.

During manual traversing in the JOG mode, the geometry axes can be traversed either in the workpiece coordinate system (WCS) or in the settable zero system (SZS).

System variables

During manual traversing in the JOG mode, the geometry axes can be traversed either in the workpiece coordinate system (WCS) or in the settable zero system (SZS). The selection is made via the system variable $AC_JOG_COORD:

$AC_JOG_COORD = <value>

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Workpiece coordinate system (WCS)</td>
</tr>
<tr>
<td>1</td>
<td>Settable zero system (SZS)</td>
</tr>
</tbody>
</table>
10.5.4.5 Suppression of frames

Suppression of frames is performed channel-specifically via the commands G53, G135 and SUPA described in the following. Activation of the frame suppression results in jumps in the position display (HMI) as well as in the position values in system variables that refer to the WCS, SZS or BZS. The behavior can be set via machine data.

Machine data

The behavior of position displays (HMI) and position values in system variables is specified via the following machine data:

MD24020 $MC_FRAME_SUPPRESS_MODE, bit<n> = <value> (positions during frame suppression)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Position values (OPI) with frame suppression ¹</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Position values (OPI) without frame suppression ²</td>
</tr>
</tbody>
</table>

¹) with frame suppression
²) without frame suppression
### Bit Value Meaning

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Position values in system variables with frame suppression ¹)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Position values in system variables without frame suppression ²)</td>
</tr>
</tbody>
</table>

¹) Jump of the position value
²) No jump of the position value

### Programming

#### Command Meaning

**G53:** Nonmodal suppression of the following frames:
- $P_{TRAFRAME}$
- $P_{PFRAME}$
- $P_{ISO4FRAME}$
- $P_{CYCFRAME}$
- $P_{IFRAME}$
- $P_{GFRAME}$
- $P_{TOOLFRAME}$
- $P_{WPFRAME}$

**G153:** Non-modal suppression of the frames as for G53 plus following frames:
- $P_{PARTFRAME}$
- $P_{SETFRAME}$
- $P_{EXTFRAME}$
- $P_{ACTBFRAME}$
- $P_{ISO1FRAME}$
- $P_{ISO2FRAME}$
- $P_{ISO3FRAME}$

**SUPA:** Implicit preprocessing stop and non-modal suppression of the frames as for G53 and G135 plus following frames:
- Handwheel offsets (DRF)
- Overlaid movement
- External work offset

**G500:** Modal activation of the G500 frame. The G500 frame should normally be a zero frame.

**DRFOF:** Deactivation (deletion) of the handwheel offsets (DRF)

### 10.5.5 Frames of the frame chain

#### 10.5.5.1 Overview

The following frames are available:
- **Settable frames** (G500, G54 ... G57, G505 ... G599)
- **Grinding frames** (GFRAME0 ... GFRAME100)
- Basic frames
- Programmable frame
- System frames
10.5.5.2 Settable frames ($P_{UIFR}[<n>]$)

**Machine data**

- **Channel-specific settable frames**
  - The number of channel-specific settable frames is set with the following machine data:
    
    $MD28080 \ $MC_{MM\ \text{NUM\ USER\ FRAMES}} = <\text{number}>$
  
  System variable index $n = 0, 1, 2, \ldots <\text{number}> - 1$

- **NCU-global settable frames**
  - The number of NCU-global settable frames is set with the following machine data:
    
    $MD18601 \ $MN_{MM\ \text{NUM\ GLOBAL\ USER\ FRAMES}} = <\text{number}>$
  
  System variable index $n = 0, 1, 2, \ldots <\text{number}> - 1$
  
  If the machine data has a value > 0, there are no channel-specific settable frames. The machine data to set the channel-specific settable frames is then not evaluated.

**System variables**

- **$P_{UIFR}[<n>]$ (settable frames of the data management)**
  - System variable $P_{UIFR}[<n>]$ can be used to read and write the settable frames of the data management. The new values are not immediately active in the channel when writing a settable frame of the data management. The activation in the channel only takes effect with the programming of a work offset $G500, \ G54...G599$.
  
  For NCU-global frames, the changed settable frame of the data management is active in each channel of the NCU which executes a $G500, \ G54..G599$ command.
  
  The settable frames in the data management are also stored during a data backup.

- **$P_{IFRAME}$ (active settable frame)**
  - System variable $P_{IFRAME}$ can be used to read and write the settable frame active in the channel. The new values are immediately active in the channel when writing the settable frame.
  
  In the case of NCU-global settable frames, the changed active frame acts only in the channel in which the new values were programmed. If the changed NCU-global settable frame is to be active in all channels of the NCU, the settable frame active in the channel and the corresponding settable frame of the data management must be written together:
    
    $P_{UIFR}[<n>] = P_{IFRAME} = <\text{new value}>$
  
  - $P_{UIFR}[<n>]$ (settable frame in the data management)
  - $P_{IFRAME}$ (settable frame active in the channel)
  
  For the changed settable frame to take effect in another channel, it must be activated in this channel with the appropriate command, e.g. $G54$. 

Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2 729
System variable $P\_UIFRNUM$ can be used to read the index $<n>$ of the settable frame of the data management active in the channel:

Settable frame active in the channel $P\_IFRAME == P\_UIFR[ P\_UIFRNUM ]$

<table>
<thead>
<tr>
<th>$P_UIFRNUM$</th>
<th>$P_IFRAME == P_UIFR[&lt;n&gt;], where n =</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>99</td>
<td>99</td>
</tr>
</tbody>
</table>

**Programming**

A channel-specific settable frame of the data management $P\_UIFR[<n>]$ becomes the settable frame $P\_IFRAME$ active in the channel through the appropriate command ($G54 \ldots G57, G505 \ldots G599$ and $G500$).

<table>
<thead>
<tr>
<th>Command</th>
<th>Activation of the settable frame of the data management</th>
</tr>
</thead>
<tbody>
<tr>
<td>G500</td>
<td>$P_UIFR[0]$</td>
</tr>
<tr>
<td>G54</td>
<td>$P_UIFR[1]$</td>
</tr>
<tr>
<td>G55</td>
<td>$P_UIFR[2]$</td>
</tr>
<tr>
<td>G56</td>
<td>$P_UIFR[3]$</td>
</tr>
<tr>
<td>G57</td>
<td>$P_UIFR[4]$</td>
</tr>
<tr>
<td>G505</td>
<td>$P_UIFR[5]$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>G599</td>
<td>$P_UIFR[99]$</td>
</tr>
</tbody>
</table>

**Supplementary conditions**

Writing of settable frames through HMI/PLC

Only the settable frames of the data management can be written from the HMI or the PLC user program.

**10.5.5.3 Grinding frames $P\_GFR[<n>]$**

Grinding frames are additional work offsets or "fit-dependent corrections" available particularly for the grinding technology. They have an additive effect on the work offsets of the settable frames (Page 729).

**Machine data**

Channel-specific grinding frames

The number of channel-specific grinding frames is set with the following machine data:
NCU-global grinding frames

The number of NCU-global grinding frames is set with the following machine data:

```
MD28079 $MN_MM_NUM_G_FRAMES = <number>
System variable index n = 0, 1, 2, ... <number> - 1
```

If the machine data has a value > 0, there are no channel-specific grinding frames. The machine data to set the channel-specific grinding frames is then not evaluated.

System variables

$P_GFR[<n>] (grinding frames of the data management)

System variable $P_GFR[n] can be used to read and write the grinding frames of the data management. The new values are not immediately active in the channel when writing a grinding frame. The activation in the channel only takes effect with the programming of the appropriate command GFRAME0...GFRAME100. For NCU-global frames, the changed frame only becomes active in those channels of the NCU, which execute a GFRAME0...GFRAME100 command.

The grinding frames in the data management are also stored during a data backup.

$P_GFRAME (active grinding frame)

System variable $P_GFRAME can be used to read and write the grinding frame active in the channel. The new values are immediately active in the channel when writing the active grinding frame.

In the case of NCU-global settable frames, the changed frame acts only in the channel in which the new frame values were programmed.

If the changed NCU-global settable frame is to be active in all channels of the NCU starting from one channel, the settable frame active in the channel and the settable frame in the data management must be written together:

```
$P_GFR[<n>] = $P_GFRAME = <new value>
```

- $P_GFR[<n>] (grinding frame of the data management)
- $P_GFRAME (grinding frame active in the channel)

For the changed grinding frame to take effect in another channel, it must be activated in this channel with the appropriate command GFRAME<n>.

$P_GFRNUM (number of the active grinding frame)

System variable $P_GFRNUM can be used to read the index <n> of the grinding frame of the data management active in the channel:

Grinding frame active in the channel $P_GFRAME == $P_GFR[ $P_GFRNUM ]

<table>
<thead>
<tr>
<th>$P_GFRNUM</th>
<th>$P_GFRAME == $P_GFR[&lt;n&gt;], where n =</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
$P_GFRNUM | $P_GFRAME == $P_GFR\[<n>\], where n = 
---|---
2  | 2
... | ...
99 | 99

**Programming**

A channel-specific grinding frame of the data management $P_GFR[<n>] becomes the grinding frame $P_GFRAME active in the channel through the appropriate command (GFRAME0 ... GFRAME100).

<table>
<thead>
<tr>
<th>Command</th>
<th>Activation of the grinding frame of the data management</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFRAME0</td>
<td>$P_GFR[ 0 ]</td>
</tr>
<tr>
<td>GFRAME1</td>
<td>$P_GFR[ 1 ]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>GFRAME100</td>
<td>$P_GFR[ 99 ]</td>
</tr>
</tbody>
</table>

**Supplementary conditions**

**Writing of grinding frames through HMI/PLC**

Only the grinding frames of the data management can be written from the HMI or the PLC user program.

**10.5.5.4 Channel-specific basic frames[<n>]**

**Machine data**

**Number of channel-specific basic frames**

The number of channel-specific basic frames is set with the following machine data:

MD28081 $MC_MM_NUM_BASE_FRAMES = <number>

System variable index n = 0, 1, 2, ... <number> - 1

**System variables**

$P_CHBFR[<n>] (channel-specific basic frames of the data management)

System variable $P_CHBFR[<n>] can be used to read and write the channel-specific basic frames of the data management. The new values are not immediately active in the channel when writing a channel-specific basic frame. The activation in the channel only takes effect with the programming of the appropriate command G500, G54...G599.

The channel-specific basic frames in the data management are also stored during a data backup.
$P_{CHBFRAME}[<n>] (active channel-specific basic frames)
System variable $P_{CHBFRAME}[<n>] can be used to read and write the active channel-specific basic frames. When writing an active channel-specific basic frame, the new values take effect immediately through the recalculation of the active complete basic frame $P_{ACTBFRAME}.

System variables for compatibility reasons

$P_{UBFR} (first channel-specific basic frames of the data management)
The system variable is retained for reasons of compatibility, although it is redundant for the $P_{CHBFR}[0]$ variables.

The basic frame with field device 0 is not activated simultaneously when writing to the predefined $P_{UBFR}$ variable, but rather activation only takes place on execution of a G500, G54, G599 command. For NCU-global frames, the changed frame only becomes active in those channels of the NCU, which execute a G500, G54..G599 command. The variable is used primarily for storing write operations to the basic frame on HMI or PLC. The variable can also be read and written in the program.

$P_{UBFR}$ is identical to $P_{CHBFR}[0]$. One basic frame always exists in the channel by default, so that the system variable is compatible with older versions. If there is no channel-specific basic frame, an alarm is issued at read/write: "Frame: Statement not permitted" is output on a read or write access.

$P_{BFRAME} (first active channel-specific basic frame)
The system variable is retained for reasons of compatibility, although it is redundant for the $P_{CHBFRAME}[0]$ variables.

The predefined frame variable $P_{BFRAME}$ can be used to read and write the current basic frame with the array index 0, which is valid in the channel, in the part program. The written basic frame is immediately included in the calculation. In the case of NCU-global settable frames, the modified frame acts only in the channel in which the frame was programmed. If the frame is to be modified for all channels of an NCU, $P_{UBFR}$ and $P_{BFRAME}$ must be written simultaneously. The other channels must then activate the corresponding frame, e.g. with G54.

$P_{BFRAME}$ is identical to $P_{CHBFRAME}[0]$. The system variable always has a valid default value. If there is no channel-specific basic frame, an alarm is issued at read/write: "Frame: Statement not permitted" is output on a read or write access.

Supplementary conditions

Writing of grinding frames through HMI/PLC
Only the grinding frames of the data management can be written from the HMI or the PLC user program.

Basic frames can be read and written via part programs, via the OPI (only data management frames and from the PLC).
10.5.5.5 NCU-global basic frames $P_NCBFR[<n>]

Machine data

Number of NCU-global basic frames
The number of NCU-global basic frames is set with the following machine data:

MD18602 $MN.MM_NUM_GLOBAL_BASE_FRAMES = <number>

System variable index n = 0, 1, 2, ..., <number> - 1

System variables

$P_NCBFR[<n>] (NCU-global basic frames of the data management)
System variable $P_NCBFR[<n>] can be used to read and write the NCU-global basic frames of the data management. The new values are not immediately active in the channel when writing an NCU-global basic frame. The activation in the channel only takes effect with the programming of the appropriate command G500, G54...G599.

The NCU-global basic frames in the data management are also stored during a data backup.

$P_NCBFRAME[<n>] (current NCU-global basic frames)
System variable $P_NCBFRAME[<n>] can be used to read and write the active NCU-global basic frames. When writing an active NCU-global basic frame, the new values take effect immediately through the recalculation of the active complete basic frame $P_ACTBFRAME.

If the changed NCU-global basic frame is to be active in all channels of the NCU starting from one channel, the NCU-global basic frame active in the channel and the NCU-global frame in the data management must be written together:

$P_NCBFR[<n>] = $P_NCBFRAME = <new value>

- $P_NCBFR[<n>] (NCU-global basic frame of the data management)
- $P_NCBFRAME (NCU-global basic frame active in the channel)

For the changed NCU-global frame to take effect in another channel, it must be activated in this channel with the appropriate command G500, G54..G599.

Programming

A channel-specific settable frame of the data management $P_UIFR[<n>] becomes the settable frame $P_IFRAME active in the channel through the appropriate command (G54 ..., G57, G505 ..., G599 and G500).

<table>
<thead>
<tr>
<th>Command</th>
<th>Activation of the NCU-global and channel-specific basic frames of the data management</th>
</tr>
</thead>
<tbody>
<tr>
<td>G500</td>
<td>$P_CHBFR[ 0 ] : $P_NCBFR[0]</td>
</tr>
</tbody>
</table>
### Command Table

<table>
<thead>
<tr>
<th>Command</th>
<th>Activation of the NCU-global and channel-specific basic frames of the data management</th>
</tr>
</thead>
<tbody>
<tr>
<td>G57</td>
<td>$P_{CHBFR}[4] : P_{NCBFR}[4]</td>
</tr>
<tr>
<td>G505</td>
<td>$P_{CHBFR}[5] : P_{NCBFR}[5]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>G599</td>
<td>$P_{CHBFR}[99] : P_{NCBFR}[99]</td>
</tr>
</tbody>
</table>

#### 10.5.5.6 Active complete basic frame $P_{ACTBFRAME}$

**Function**

All active NCU-global and channel-specific basic frames are combined into the complete basic frame $P_{ACTBFRAME}$:

\[
P_{ACTBFRAME} = P_{NCBFRAME}[0] : \ldots : P_{NCBFRAME}[<n>] : P_{CHBFRAME}[0] : \ldots : P_{CHBFRAME}[<n>]
\]

![Complete basic frame](image)

**Figure 10-22 Complete basic frame**

**Machine data**

**Reset response**

Which basic frames are active after a reset (channel reset, end of program reset or Power On) is set via the machine data:
MD20110 $MC_RESET_MODE_MASK, bit0 = 1 and bit14 = 1

- Bit 1 = 0: Default value ⇒ reset behavior corresponding to the setting of the further bits
- Bit14 = 0: The basic frames are completely deselected with reset.
- Bit14 = 1: With reset, the machine data settings are taken over in the system variables and the basic frames selected therein become active:
  - $P_NCBFRMASK = MD10613 $MN_NCBFRAME_RESET_MASK
  - $P_CHBFRMASK = MD24002 $MC_CHBFRAME_RESET_MASK

**Example**

<table>
<thead>
<tr>
<th>Machine data setting</th>
<th>Active basic frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_NCBFRMASK = MD10613 $MN_NCBFRAME_RESET_MASK = 'H81'</td>
<td>$P_NCBFRAME[0] : $P_NCBFRAME[7]</td>
</tr>
</tbody>
</table>

**Programming**

**Basic frame masks**

The basic frames that are linked to form the complete basic frame are selected via the basic frame masks $P_NCBFRMASK$ and $P_CHBFRMASK$.

The appropriate basic frame is selected by setting a bit in the basic frame mask:

- $P_NCBFRMASK, \text{bit } 0, 1, 2, \ldots \ n \Rightarrow P_NCBFRAME[0, 1, 2, \ldots \ n]$
- $P_CHBFRMASK, \text{bit } 0, 1, 2, \ldots \ n \Rightarrow P_NCHFRAME[0, 1, 2, \ldots \ n]$

The basic frame masks $P_NCBFRMASK$ and $P_CHBFRMASK$ can only be read/written in the NC program. The basic frame masks can be read via the OPI.

After writing a basic frame mask, the active complete basic frame $P_ACTBFRAME$ and complete frame $P_ACTFRAME$ are recalculated.

**Example**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_NCBFRMASK = 'H81'$</td>
<td>; Active NCU-global basic frames: $P_NCBFRAME[0]$ : $P_NCBFRAME[7]$</td>
</tr>
</tbody>
</table>

**10.5.5.7 Programmable frame $P_PFRAME$**

Programmable frames are available only as active frames.

This frame is reserved for the programmer.

The programmable frame remains at RESET, if:

MD24010 $MC_PFRAME_RESET_MODE \text{ (reset mode for programmable frame)} = 1$

This functionality is especially important after a RESET if one still wants to retract out of an oblique hole.
MIRROR

Mirrorings of a geometry axis were previously (up to SW-P4) related to a defined reference axis only using the machine data:

\[ \text{MD10610 } \text{$MN\_MIRROR\_REF\_AX$} \]

(reference axis for the mirroring).

From the user's point of view, this definition is difficult to understand. When mirroring the z axis, the display showed that the x axis was mirrored and the y axis had been rotated through 180 degrees. When mirroring two axes this became even more complex and it was no longer easy to understand, which axes had been mirrored and, which had not.

As of SW P5, there is the option to clearly display the mirroring of an axis. Mirroring is then not mapped to mirroring of a reference axis and rotations of other axes.

This setting can be configured using:

\[ \text{MD10610 } \text{$MN\_MIRROR\_REF\_AX = 0$} \]

\text{MIRROR and AMIRROR are used to expand the programming of the programmable frame. Previously, the specified value of the coordinate axis, e.g. the value 0 for MIRROR X0 was not evaluated, but rather the AMIRROR had a toggle function, i.e. MIRROR X0 activates mirroring and an additional AMIRROR X0 deactivates it. MIRROR always has an absolute effect and AMIRROR an additive effect.}

The

\[ \text{MD10612 } \text{$MN\_MIRROR\_TOGGLE = 0$} \]

("Mirror Toggle")

machine data setting can be used to define that the programmed values are evaluated. A value of 0, as in AMIRROR X0, deactivates the mirroring of the axis, and values not equal to 0 cause the axis to be mirrored if it is not already mirrored.

Reading or writing mirroring component-by-component is independent of the machine data:

\[ \text{MD10612 } \text{$MN\_MIRROR\_TOGGLE$} \]

A value = 0 means that the axis is not mirrored and a value = 1 means that the axis will always be mirrored, irrespective of whether it has already been mirrored or not.

\[
\begin{align*}
\text{P\_NCBF[0,x,mi]=1} & \quad ; \text{x axis is always mirrored.} \\
\text{P\_NCBF[0,x,mi]=0} & \quad ; \text{x axis mirroring is OFF.}
\end{align*}
\]

Axis-specific replacement G58, G59 (only 840D sl)

The translation component of the programmable frame is split into an absolute component and a component for the total of all additively programmed translations. The absolute component can be changed using TRANS, CTRANS or by writing the translation components, in which the additive component is set to zero. G58 changes only the absolute translation component for the specified axis; the total of additively programmed translations is retained.

\[
\begin{align*}
\text{G58 X... Y... Z... A... ...} \\
\text{G59 is used for axis-specific overwriting of the additively programmed translations for the specified axes which were programmed with ATRANS.}
\end{align*}
\]

\[
\begin{align*}
\text{G59 X... Y... Z... A... ...}
\end{align*}
\]
Example

TRANS X10 Y10 Z10
ATRANS X5 Y5 ; Total translations X15 Y15 Z10
G58 X20 ; Total translations X25 Y15 Z10
G59 X10 Y10 ; Total translations X30 Y20 Z10

G58 and G59 can only be used if:

MD24000 $MC_FRAME_ADD_COMPONENTS (frame components for G58 / G59) == TRUE

The table below shows the effect of various program commands on the absolute and additive translation.

<table>
<thead>
<tr>
<th>Command</th>
<th>Coarse or absolute offset</th>
<th>Fine or additive offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANS X10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ATRANS X10</td>
<td>Unchanged</td>
<td>alt_fine + 10</td>
</tr>
<tr>
<td>CTRANS(X,10)</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>CTRANS()</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CFINE(X,10)</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>$P_PFRAME[X,TR] = 10</td>
<td>10</td>
<td>Unchanged</td>
</tr>
<tr>
<td>$P_PFRAME[X,FI] = 10</td>
<td>Unchanged</td>
<td>10</td>
</tr>
<tr>
<td>G58 X10</td>
<td>10</td>
<td>Unchanged</td>
</tr>
<tr>
<td>G59 X10</td>
<td>Unchanged</td>
<td>10</td>
</tr>
</tbody>
</table>

10.5.8 Channelspecific system frames

Channel-specific system frames are only written by system functions, such as actual value setting, scratching, external work offset and inclined machining.

Machine data

Parameterization of the channel-specific system frames

Only channel-specific system frames whose system functions are actually used should be configured for memory space reasons.

Per channel, each system frame occupies approx. 1 KB static and approx. 6 KB dynamic memory.

The channel-specific system frames are parameterized in the following machine data:

MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit<n>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>System frame available:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>$P_SETFR: Actual value setting and scratching</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$P_EXTFR: External work offset via system frames</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>$P_PARTFR: TCARR and PAROT with an orientable toolholder</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>$P_TOOLFR: TOROT and TOFRAME</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>$P_WPFR: Frame for workpiece reference points</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>$P_CYCFR: Frame for cycles</td>
</tr>
</tbody>
</table>
Parameterization of the SZS (ACS) coordinate system

The following machine data is used to specify which system frames form the SZS (ACS) coordinate system

MD24030 \$MC_FRAME_ACS_SET = <value>

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning: The SZS (ACS) coordinate system comprises</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$P_{PARTFRAME}$ : $P_{SETFRAME}$ : $P_{EXTFRAME}$ : $P_{ISO1FRAME}$ : $P_{ISO2FRAME}$ : $P_{ISO3FRAME}$ : $P_{ACTBFRAME}$ : $P_{IFRAME}$ : $P_{TOOLFRAME}$ : $P_{WPFRAME}$</td>
</tr>
<tr>
<td>1</td>
<td>$P_{PARTFRAME}$ : $P_{SETFRAME}$ : $P_{EXTFRAME}$ : $P_{ISO1FRAME}$ : $P_{ISO2FRAME}$ : $P_{ISO3FRAME}$ : $P_{ACTBFRAME}$ : $P_{IFRAME}$ : $P_{TOOLFRAME}$ : $P_{WPFRAME}$ : $P_{TRAFRAME}$ : $P_{PFRAME}$ : $P_{ISO4FRAME}$</td>
</tr>
</tbody>
</table>

System variables

Channel-specific system frames of the data management

The channel-specific system frames of the data management can be read and written via the following frame variables:

<table>
<thead>
<tr>
<th>System variable</th>
<th>Meaning: System frame of the data management for</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{SETFR}$</td>
<td>Actual value setting and scratching (Set Frame)</td>
</tr>
<tr>
<td>$P_{EXTFR}$</td>
<td>External work offset (Ext Frame)</td>
</tr>
<tr>
<td>$P_{PARTFR}$</td>
<td>$TCARR$ and $PAROT$ for orientable toolholder (Part Frame)</td>
</tr>
<tr>
<td>$P_{TOOLFR}$</td>
<td>$TOROT$ and $TOFRAME$ (Tool Frame)</td>
</tr>
<tr>
<td>$P_{WPFR}$</td>
<td>Workpiece reference points (WorkPiece Frame)</td>
</tr>
<tr>
<td>$P_{CYCFR}$</td>
<td>Cycles (Cycle Frame)</td>
</tr>
<tr>
<td>$P_{TRAFRAME}$</td>
<td>Transformations (Transformation Frame)</td>
</tr>
<tr>
<td>$P_{ISO1FR}$</td>
<td>G51.1 mirroring (ISO)</td>
</tr>
<tr>
<td>$P_{ISO2FR}$</td>
<td>G68 2DROT (ISO)</td>
</tr>
<tr>
<td>$P_{ISO3FR}$</td>
<td>G68 3DROT (ISO)</td>
</tr>
<tr>
<td>$P_{ISO4FR}$</td>
<td>G51 scale (ISO)</td>
</tr>
<tr>
<td>$P_{RELFR}$</td>
<td>Relative coordinate systems</td>
</tr>
</tbody>
</table>
Note

Cycle programming

The frame variables of the system frames are only for the cycle programming. Therefore, in NC programs the system frames should not be written directly by the user, but rather only via system functions such as TOROT, PAROT, etc.

Channelspecific active system frames

System variables of the active channel-specific system frames:

<table>
<thead>
<tr>
<th>System variable</th>
<th>Meaning: Active system frame for</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_SETFRAME</td>
<td>Actual value setting and scratching (Set Frame)</td>
</tr>
<tr>
<td>$P_EXTFRAME</td>
<td>External work offset (Ext Frame)</td>
</tr>
<tr>
<td>$P_PARTFRAME</td>
<td>TCARR and PAROT for orientable toolholder (Part Frame)</td>
</tr>
<tr>
<td>$P_TOOLFRAME</td>
<td>TOROT and TOFRAME (Tool Frame)</td>
</tr>
<tr>
<td>$P_WPFRAME</td>
<td>Workpiece reference points (WorkPiece Frame)</td>
</tr>
<tr>
<td>$P_CYCFRAME</td>
<td>Cycles (Cycle Frame)</td>
</tr>
<tr>
<td>$P_TRAFRAME</td>
<td>Transformations (Transformation Frame)</td>
</tr>
<tr>
<td>$P_ISO1FRAME</td>
<td>G51.1 mirroring (ISO)</td>
</tr>
<tr>
<td>$P_ISO2FRAME</td>
<td>G68 2DROT (ISO)</td>
</tr>
<tr>
<td>$P_ISO3FRAME</td>
<td>G68 3DROT (ISO)</td>
</tr>
<tr>
<td>$P_ISO4FRAME</td>
<td>G51 scale (ISO)</td>
</tr>
<tr>
<td>$P_RELFRAME</td>
<td>Relative coordinate systems</td>
</tr>
</tbody>
</table>

If a channel-specific system frame of the data management is not parameterized, the following applies for the corresponding active system frame: $P_<system frame> == zero frame.

Channel-specific active SZS (ACS) complete frame

The system variable $P_ACSFRAME is used to read which system frames form the SZS (ACS) coordinate system. The specification is made via the machine data described above. See Subsection "Machine data" > "Parameterization of the SZS (ACS) coordinate system"
Four modes can be set via machine data:

- **MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 0**
  The current complete frame is deleted when geometry axes are switched over, when transformations are selected and deselected, and on GEOAX(). The modified geometry axis configuration is not used until a new frame is activated.

- **MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1**
  The current complete frame is calculated again when the geometry axes are switched over, and the translations, scalings and mirrorings of the new geometry axes are effective. The rotations of the geometry axes which were programmed before the switchover remain effective for the new geometry axes.
  For TRANSMIT, TRACYL and TRAANG, see Section "Frame for selection and deselection of transformations (Page 743)".

- **MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2**
  The current complete frame is calculated again when the geometry axes are switched over, and the translations, scalings and mirrorings of the new geometry axes are effective. If rotations are active in the current basic frames, the current settable frame or the programmable frame before the switchover, it is aborted with the alarm "Frame: Geometry axis switchover not allowed".
  For TRANSMIT, TRACYL and TRAANG, see Section "Frame for selection and deselection of transformations (Page 743)".

- **MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 3**
  The current frame is deleted when selecting and deselecting transformations.
  With GEOAX(), the current complete frame is calculated again and the translations, scalings and mirrorings of the new geometry axes come into effect.
  The rotations of the geometry axes which were programmed before the switchover remain effective for the new geometry axes.

The workpiece geometry is described by a coordinate system that is formed by the geometry axes. A channel axis is assigned to each geometry axis and a machine axis is assigned to each channel axis. An axial frame exists for each machine axis and for each frame (system frame, basic frame, settable frame, programmable frame). When a new machine axis is assigned to a geometry axis, the axial frame components of the machine axis, such as translations (coarse and fine), scaling and mirroring of the appropriate frame, are also applied. The new geometry in the channel is then generated by the new contour frames resulting from the new geometry axes (up to three in number).

The current valid frames are calculated again on the geometry axis switchover and a resulting complete frame is generated. The data management frames are not included unless they are activated.

**Example:**

The channel axis is to become a geometry axis x through geo axis substitution. The substitution is to give the programmable frame a translation component of 10 in the x axis. The current settable frame is to be retained.

**MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_UIFR[1] = CROT(x,10,y,20,z,30)</td>
<td>Frame is retained after geo axis substitution.</td>
</tr>
<tr>
<td>G54</td>
<td>Settable frame becomes active.</td>
</tr>
<tr>
<td>TRANS a10</td>
<td>Axial offset of a is also substituted.</td>
</tr>
</tbody>
</table>
Program code

```
GEOAX(1, a)
; a becomes x axis;

$P_ACTFRAME=CROT(x, 10, y, 20, z, 30):CTRANS(x10).
```

Several channel axes can become geometry axes on a transformation change.

Example:

Channel axes 4, 5, and 6 become the geometry axes of a 5-axis transformation. The geometry axes are thus all substituted before the transformation. The current frames are changed when the transformation is activated. The axial frame components of the channel axes which become geometry axes are taken into account when calculating the new WCS. Rotations programmed before the transformation are retained. The old WCS is restored when the transformation is deactivated. The most common application will be that the geometry axes do not change before and after the transformation and that the frames are to stay as they were before the transformation.

**Machine data:**

```
$MN_FRAME_GEOAX_CHANGE_MODE = 1

$MC_AXCONF_CHANAX_NAME_TAB[0] = "CAX"
$MC_AXCONF_CHANAX_NAME_TAB[1] = "CAY"
$MC_AXCONF_CHANAX_NAME_TAB[2] = "CAZ"
$MC_AXCONF_CHANAX_NAME_TAB[3] = "A"
$MC_AXCONF_CHANAX_NAME_TAB[4] = "B"
$MC_AXCONF_CHANAX_NAME_TAB[5] = "C"

$MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1
$MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 2
$MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3

$MC_AXCONF_GEOAX_NAME_TAB[0] = "X"
$MC_AXCONF_GEOAX_NAME_TAB[1] = "Y"
$MC_AXCONF_GEOAX_NAME_TAB[2] = "Z"

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0] = 4
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1] = 5
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2] = 6

$MC_TRAFO_AXES_IN_1[0] = 4
$MC_TRAFO_AXES_IN_1[1] = 5
$MC_TRAFO_AXES_IN_1[2] = 6
$MC_TRAFO_AXES_IN_1[3] = 1
$MC_TRAFO_AXES_IN_1[4] = 2
```
10.5.6.2 Frame for selection and deselection of transformations

This function is available with NCK 51.00.00 and higher. Transformations TRANSMIT, TRACYL and TRAANG are supported.

As a rule, the assignment of geometry axes to channel axes changes when selecting and deselecting transformations. It is not possible to uniquely assign axial frame components to geometric contour frame components when carrying out transformations, in which rotary axes become linear axes and vice versa. The contour frame must be conditioned using special treatment for such non-linear transformations.

The mode, set with MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1 and 2, is expanded in such a way as to take the above transformations into account.

When selecting transformations, the contour frame is connected to the axial frames. With transformations TRANSMIT, TRACYL and TRAANG, the virtual geometry axis is subject to special treatment.

**Note**

Transformations with virtual axes

When selecting TRANSMIT or TRACYL, offsets, scaling and mirroring of the real Y axis are not transferred and accepted in the virtual Y axis. Offsets, scaling and mirroring of the virtual Y axis are deleted for TRAFOOF.
Transmit expansions:
The machine data below can be used to take the axial complete frame of the TRANSMIT rotary axis, i.e., the translation, fine offset, mirroring and scaling, into account in the transformation:

MD24905 $MC_TRANSMIT_ROT_AX_FRAME_1 = 1
MD24955 $MC_TRANSMIT_ROT_AX_FRAME_2 = 1

A rotary axis offset can, for example, be entered by compensating the oblique position of a workpiece in a frame within a frame chain. As a rule, this offset can also be included in the transformation as an offset in the rotary axis. A c axis offset, as in the figure above, then leads to corresponding x and y values.

MD24905 $MC_TRANSMIT_ROT_AX_FRAME_1 = 2
MD24955 $MC_TRANSMIT_ROT_AX_FRAME_2 = 2

With this setting, the axial offset of the rotary axis is taken account of in the transformation up to the SZS. The axial offsets of the rotary axis included in the SZS frames are entered into the transformation frame as rotation. This setting is only effective if the transformation frame has been configured.

Frame expansions:
The expansions described below are only valid for the following machine data settings:

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1
MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

The selection of transformation TRANSMIT produces a virtual geometry axis, coupled by way of the rotary axis, which is merely included in the contour frame but does not have a reference to an axial frame. The geometric value results from the rotation of a rotary axis. All other geometry axes accept their axial components when the transformation is selected.

Translations:
When selecting TRANSMIT, translations of the virtual axis are deleted. Translations of the rotary axis can be taken into account in the transformation.

**Rotations:**
Rotation before the transformation is taken over.

**Mirroring:**
Mirroring of the virtual axis is deleted. Mirroring of the rotary axis can be taken into account in the transformation.

**Scaling:**
Scaling of the virtual axis is deleted. Scaling of the rotary axis can be taken into account in the transformation.

**Example:**

Machine data for TRANSMIT

```plaintext
; FRAME configurations

$MC_MM_SYSTEM_FRAME_MASK='H41'
$MC_CHSFRAME_RESET_MASK='H41'
$MC_CHSFRAME_POWERON_MASK='H41'

$MN_FRAME_GEOAX_CHANGE_MODE=1

$MC_RESET_MODE_MASK='H4041'

$MC_GCODE_RESET_VALUES[7]=2
$MC_GCODE_RESET_VALUES[7]=1

$MN_MM_NUM_GLOBAL_USER_FRAMES=0
$MN_MM_NUM_GLOBAL_BASE_FRAMES=3

$MC_MM_NUM_USER_FRAMES=10
$MC_MM_NUM_BASE_FRAMES=3

$MN_MIRROR_REF_AX=0
$MN_MIRROR_TOGGLE=0

$MN_MM_FRAME_FINE_TRANS=1
$MC_FRAME_ADD_COMPONENTS=TRUE

; TRANSMIT is 1st transformer
```
$MC\_TRAFO\_TYPE\_1=256$

$MC\_TRAFO\_AXES\_IN\_1[0]=1$
$MC\_TRAFO\_AXES\_IN\_1[1]=6$
$MC\_TRAFO\_AXES\_IN\_1[2]=3$
$MC\_TRAFO\_AXES\_IN\_1[3]=0$
$MC\_TRAFO\_AXES\_IN\_1[4]=0$

$MA\_ROT\_IS\_MODULO\{AX6\}=TRUE;$

$MC\_TRAFO\_GEOAX\_ASSIGN\_TAB\_1[0]=1$
$MC\_TRAFO\_GEOAX\_ASSIGN\_TAB\_1[1]=6$
$MC\_TRAFO\_GEOAX\_ASSIGN\_TAB\_1[2]=3$

$MC\_TRANSMIT\_BASE\_TOOL\_1[0]=0.0$
$MC\_TRANSMIT\_BASE\_TOOL\_1[1]=0.0$
$MC\_TRANSMIT\_BASE\_TOOL\_1[2]=0.0$

$MC\_TRANSMIT\_ROT\_AX\_OFFSET\_1=0.0$
$MC\_TRANSMIT\_ROT\_SIGN\_IS\_PLUS\_1=TRUE$

$MC\_TRANSMIT\_ROT\_AX\_FRAME\_1=1$

; TRANSMIT is 2nd transformer

$MC\_TRAFO\_TYPE\_2=256$

$MC\_TRAFO\_AXES\_IN\_2[0]=1$
$MC\_TRAFO\_AXES\_IN\_2[1]=6$
$MC\_TRAFO\_AXES\_IN\_2[2]=2$
$MC\_TRAFO\_AXES\_IN\_2[3]=0$
$MC\_TRAFO\_AXES\_IN\_2[4]=0$

$MC\_TRAFO\_GEOAX\_ASSIGN\_TAB\_2[0]=1$
$MC\_TRAFO\_GEOAX\_ASSIGN\_TAB\_2[1]=6$
$MC\_TRAFO\_GEOAX\_ASSIGN\_TAB\_2[2]=2$

$MC\_TRANSMIT\_BASE\_TOOL\_2[0]=4.0$
$MC\_TRANSMIT\_BASE\_TOOL\_2[1]=0.0$
$MC\_TRANSMIT\_BASE\_TOOL\_2[2]=0.0$

$MC\_TRANSMIT\_ROT\_AX\_OFFSET\_2=19.0$
$MC\_TRANSMIT\_ROT\_SIGN\_IS\_PLUS\_2=TRUE$
Part program:

; Frame settings
N820 $P_UIFR[1] = ctrans(x,1,y,2,z,3,c,4)
N830 $P_UIFR[1] = $P_UIFR[1] : crot(x,10,y,20,z,30)
N850
N860 $P_CHBFR[0] = ctrans(x,10,y,20,z,30,c,15)
N870

; Tool selection, clamping compensation, plane selection
N890 T2 D1 G54 G17 G90 F5000 G64 SOFT
N900

; Approach start position
N920 G0 X20 Z10
N930
N940 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,C,15)
N950 setal(61000)
N960 endif
N970 if $P_BFRAME <> $P_CHBFR[0]
N980 setal(61000)
N990 endif
N1000 if $P_IFRAME <> CTRANS(X,1,Y,2,Z,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1010 setal(61000)
N1020 endif
N1030 if $P_IFRAME <> $P_UIFR[1]
N1040 setal(61000)
N1050 endif
N1070 setal(61000)
N1080 endif
N1090
N1100 TRANSMIT(2)
N1110
N1120 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,20,CAZ,30,C,15)
N1130 setal(61000)
N1140 endif
N1150 if $P_IFRAME <>
CTRANS(X,1,Y,0,Z,2,CAZ,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1190 setal(61000)
N1200 endif
N1240 if SP_ACTFRAME <>
  CTRANS(X,11,Y,0,Z,22,CAZ,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1250 setal(61001)
N1260 endif
N1270
N1280
N1290 $P_UIFR[1,x,tr] = 11
N1300 $P_UIFR[1,y,tr] = 14
N1310
N1320 g54
N1330

; Set frame
N1350 ROT RPL=-45
N1360 ATRANS X-2 Y10
N1370

; Four-edge roughing
N1390 G1 X10 Y-10 G41 OFFN=1; allowance 1 mm
N1400 X-10
N1410 Y10
N1420 X10
N1430 Y-10
N1440

; Tool change
N1460 G0 Z20 G40 OFFN=0
N1470 T3 D1 X15 Y-15
N1480 Z10 G41
N1490

; Four-edge finishing
N1510 G1 X10 Y-10
N1520 X-10
N1530 Y10
N1540 X10
N1550 Y-10
N1560

; Deselect frame
N2950 m30 N1580 Z20 G40
N1590 TRANS
N1600
N1610 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,20,CAZ,30,C,15)
N1620 setal(61000)
N1630 endif
N1640 if $P_BFRAME <> $P_CHBFR[0]
N1650 setal(61000)
N1660 endif
N1670 if $P_IFRAME <>
TRANS(X,11,Y,0,Z,2,CAZ,3,C,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1680 setal(61000)
N1690 endif
N1730 if $P_ACTFRAME <>
TRANS(X,21,Y,0,Z,22,CAZ,33,C,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1740 setal(61001)
N1750 endif
N1760
N1770 TRAFOOF
N1780
N1790 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,C,15)
N1800 setal(61000)
N1810 endif
N1820 if $P_BFRAME <> $P_CHBFR[0]
N1830 setal(61000)
N1840 endif
N1850 if $P_IFRAME <>
N1860 setal(61000)
N1870 endif
N1880 if $P_IFRAME <> $P_UIFR[1]
N1890 setal(61000)
N1900 endif
N1910 if $P_ACTFRAME <>
N1920 setal(61002)
N1930 endif
N1940
N2010 $P_UIFR[1] = ctrans()
N2011 $P_CHBFR[0] = ctrans()
N2020 $P_UIFR[1] = ctrans(x,1,y,2,z,3,c,0)
N2021 G54
N2021 G0 X20 Y0 Z10 C0
N2030 TRANSMIT(1)
N2040 TRANS x10 y20 z30
N2041 ATRANS y200
TRACYL

TRACYL expansions:

The machine data below can be used to take the axial complete frame of the TRACYL rotary axis, i.e., the translation, fine offset, mirroring and scaling, into account in the transformation:

MD24805 $MC_TRACYL_ROT_AX_FRAME_1 = 1
MD24855 $MC_TRACYL_ROT_AX_FRAME_2 = 1

A rotary axis offset can, for example, be entered by compensating the oblique position of a workpiece in a frame within a frame chain. As a rule, this offset can also be included in the transformation as an offset in the rotary axis or as a y offset. A c axis offset, as in the figure above, then leads to corresponding x and y values.

MD24805 $MC_TRACYL_ROT_AX_FRAME_1 = 2
With this setting, the axial offset of the rotary axis is taken account of in the transformation up to the SZS. The axial offsets of the rotary axis included in the SZS frames are entered into the transformation frame as offsets on the peripheral surface. This setting is only effective if the transformation frame has been configured.

**Frame expansions:**

The expansions described below are only valid for the following machine data settings:

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1

MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

The selection of transformation TRACYL produces a virtual geometry axis on the peripheral surface, coupled via the rotary axis, which is only taken into account in the contour frame but does not have a reference to an axial frame. All components of the virtual geometry axis are deleted. All other geometry axes accept their axial components when the transformation is selected.

**Translations:**

When selecting TRACYL, translations of the virtual axis are deleted. Translations of the rotary axis can be taken into account in the transformation.

**Rotations:**

Rotation before the transformation is taken over.

**Mirroring:**

Mirroring of the virtual axis is deleted. Mirroring of the rotary axis can be taken into account in the transformation.

**Scaling:**

Scaling of the virtual axis is deleted. Scaling of the rotary axis can be taken into account in the transformation.

**Example:**

**Machine data for TRACYL:**

```plaintext
; FRAME configurations

$MC_MM_SYSTEM_FRAME_MASK = 'H41' ; TRAFRAME, SETFRAME
$MC_CHSFRAME_RESET_MASK = 'H41' ; Frames are active after Reset.
$MC_CHSFRAME_POWERON_MASK = 'H41' ; Frames are deleted for Power On.

$MN_FRAME_GEOAX_CHANGE_MODE = 1 ; Frames are calculated after switchover of the geo axis.

$MC_RESET_MODE_MASK = 'H4041' ; Basic frame is not deselected after Reset.
$MC_RESET_MODE_MASK = 'H41' ; Basic frame is deselected after Reset.

$MC_GCODE_RESET_VALUES[7] = 2 ; G54 is the default setting.
$MC_GCODE_RESET_VALUES[7] = 1 ; G500 is the default setting.
```
$$MN\_MM\_NUM\_GLOBAL\_USER\_FRAMES = 0$$
$$MN\_MM\_NUM\_GLOBAL\_BASE\_FRAMES = 3$$

$$MC\_MM\_NUM\_USER\_FRAMES = 10$$  ; from 5 to 100
$$MC\_MM\_NUM\_BASE\_FRAMES = 3$$  ; from 0 to 8

$$MN\_NCBF\_FRAME\_RESET\_MASK = 'HFF'$$
$$MC\_CHBF\_FRAME\_RESET\_MASK = 'HFF'$$

$$MN\_MIRROR\_REF\_AX = 0$$  ; No scaling when mirroring.
$$MN\_MIRROR\_TOGGLE = 0$$
$$MN\_MM\_FRAME\_FINE\_TRANS = 1$$  ; Fine offset
$$MC\_FRAME\_ADD\_COMPONENTS = TRUE$$  ; G58, G59 is possible

```
; TRACYL with groove side offset is 3rd transformer

$$MC\_TRAFO\_TYPE\_3 = 513; TRACYL$$

$$MC\_TRAFO\_AXES\_IN\_3[0] = 1$$
$$MC\_TRAFO\_AXES\_IN\_3[1] = 5$$
$$MC\_TRAFO\_AXES\_IN\_3[2] = 3$$
$$MC\_TRAFO\_AXES\_IN\_3[3] = 2$$

$$MC\_TRAFO\_GEOAX\_ASSIGN\_TAB\_3[0] = 1$$
$$MC\_TRAFO\_GEOAX\_ASSIGN\_TAB\_3[1] = 5$$
$$MC\_TRAFO\_GEOAX\_ASSIGN\_TAB\_3[2] = 3$$

$$MC\_TRACYL\_BASE\_TOOL\_1[0] = 0.0$$
$$MC\_TRACYL\_BASE\_TOOL\_1[1] = 0.0$$
$$MC\_TRACYL\_BASE\_TOOL\_1[2] = 0.0$$

$$MC\_TRACYL\_ROT\_AX\_OFFSET\_1 = 0.0$$
$$MC\_TRACYL\_ROT\_SIGN\_IS\_PLUS\_1 = TRUE$$

$$MC\_TRACYL\_ROT\_AX\_FRAME\_1 = 1$$

Part program:

; Simple traversing test with groove side offset
N450 G603
N460

; Frame settings
N500 $P\_UIFR[1] = ctrans(x,1,y,2,z,3,b,4)
N530
N540 $P\_CHBFR[0] = ctrans(x,10,y,20,z,30,b,15)
N550
N560 G54
N570

; Continuous-path mode with selected smoothing
N590 G0 x0 y0 z-10 b0 G90 F50000 T1 D1 G19 G641 ADIS=1 ADISPOS=5
N600
N610 if $P\_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,15)
N620 setal(61000)
N630 endif
N640 if $P\_BFRAME <> $P\_CHBFR[0]
N650 setal(61000)
N660 endif
N670 if $P\_IFRAME <>
TRANS(X,1,Y,2,Z,3,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N680 setal(61000)
N690 endif
N700 if $P\_IFRAME <> $P\_UIFR[1]
N710 setal(61000)
N720 endif
N730 if $P\_ACTFRAME <>
N740 setal(61000)
N750 endif
N760

; Transformation ON
N780 TRACYL(40.)
N790
N800 if $P\_BFRAME <> CTRANS(X,10,Y,0,Z,30,CAY,20,B,15)
N810 setal(61000)
N820 endif
N830 if $P\_CHBFR[0] <> CTRANS(X,10,Y,0,Z,30,CAY,20,B,15)
N840 setal(61000)
N850 endif
N860 if $P\_IFRAME <>
TRANS(X,1,Y,0,Z,3,CAY,2,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N870 setal(61000)
N880 endif
N890 if $P\_UIFR[1] <>
TRANS(X,1,Y,0,Z,3,CAY,2,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N900 setal(61000)
N910 endif
N920 if $P_ACTFRAME <>
TRANS(X,11,Y,0,Z,33,CAY,22,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N930 setal(61001)
N940 endif
N950
N960 $P_UIFR[1,x,tr] = 11
N970 $P_UIFR[1,y,tr] = 14
N980
N990 g54
N1000
N1010 if $P_BFRAME <> CTRANS(X,10,Y,0,Z,30,CAY,20,B,15)
N1020 setal(61000)
N1030 endif
N1040 if $P_BFRAME <> $P_CHBFR[0]
N1050 setal(61000)
N1060 endif
N1070 if $P_IFRAME <>
TRANS(X,11,Y,2,Z,3,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1080 setal(61000)
N1090 endif
N1100 if $P_IFRAME <> $P_UIFR[1]
N1110 setal(61000)
N1120 endif
N1130 if $P_ACTFRAME <>
TRANS(X,21,Y,0,Z,33,CAY,22,B,19):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1140 setal(61001)
N1150 endif
N1160

; Transformation off
N1180 TRAFOOF
N1190
N1200 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,15)
N1210 setal(61000)
N1220 endif
N1230 if $P_BFRAME <> $P_CHBFR[0]
N1240 setal(61000)
N1250 endif
N1260 if $P_IFRAME <>
TRANS(X,11,Y,2,Z,3,B,4):CROT(X,10,Y,20,Z,30):CMIRROR(X,B)
N1270 setal(61000)
N1280 endif
TRAANG

Frame expansions:
The expansions described below are only valid for the following machine data settings:
MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 1
MD10602 $MN_FRAME_GEOAX_CHANGE_MODE = 2

Translations:
When selecting TRAANG, translations of the virtual axis are deleted.

Rotations:
Rotation before the transformation is taken over.

Mirroring:
Mirrorings of the virtual axis are taken over.

Scaling:
Scalings of the virtual axis are taken over.

**Example:**

Machine data for TRAANG:

```plaintext
; FRAME configurations

$MC_MM_SYSTEM_FRAME_MASK = 'H1' ; SETFRAME
$MC_CHSFRAME_RESET_MASK = 'H41' ; Frames are active after RESET.
$MC_CHSFRAME_POWERON_MASK = 'H41' ; Frames are deleted for "Power On".
$MN_FRAME_GEOAX_CHANGE_MODE = 1 ; Frames are calculated after switchover of the geo axis.

$MC_RESET_MODE_MASK = 'H4041' ; Basic frame is not deselected after RESET.
;$MC_RESET_MODE_MASK = 'H41' ; Basic frame is deselected after RESET.
;$MC_GCODE_RESET_VALUES[7] = 2 ; G54 is the default setting.
$MC_GCODE_RESET_VALUES[7] = 1 ; G500 is the default setting.

$MN_MM_NUM_GLOBAL_USER_FRAMES = 0
$MN_MM_NUM_GLOBAL_BASE_FRAMES = 3

$MC_MM_NUM_USER_FRAMES = 10 ; from 5 to 100
$MC_MM_NUM_BASE_FRAMES = 3 ; from 0 to 8

$MN_NCBFRAME_RESET_MASK = 'HFF'
$MC_CHBFRAME_RESET_MASK = 'HFF'

$MN_MIRROR_REF_AX = 0 ; No scaling when mirroring.
$MN_MIRROR_TOGGLE = 0
$MN_MM_FRAME_FINE_TRANS = 1 ; Fine offset
$MC_FRAME_ADD_COMPONENTS = TRUE ; G58, G59 is possible.

; TRAANG is 1st transformer

$MC_TRAFO_TYPE_1 = 1024

$MC_TRAFO_AXES_IN_1[0] = 4 ; Inclined axis
$MC_TRAFO_AXES_IN_1[1] = 3 ; Axis is parallel to z
$MC_TRAFO_AXES_IN_1[2] = 2
$MC_TRAFO_AXES_IN_1[3] = 0
$MC_TRAFO_AXES_IN_1[4] = 0

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=4
```
$MC.TRAFO.GEOAX.ASSIGN.TAB.1[1] = 2
$MC.TRAFO.GEOAX.ASSIGN.TAB.1[2] = 3

$MC.TRAANG.ANGLE.1 = 85.
$MC.TRAANG.PARALLEL.VELO.RES.1 = 0.
$MC.TRAANG.PARALLEL.ACCEL.RES.1 = 0.

$MC.TRAANG.BASE.TOOL.1[0] = 0.0
$MC.TRAANG.BASE.TOOL.1[1] = 0.0
$MC.TRAANG.BASE.TOOL.1[2] = 0.0

; TRAANG is 2nd transformer

$MC.TRAFO_TYPE.2 = 1024

$MC.TRAFO_AXES.IN.2[0] = 4
$MC.TRAFO_AXES.IN.2[1] = 3
$MC.TRAFO_AXES.IN.2[2] = 0
$MC.TRAFO_AXES.IN.2[3] = 0
$MC.TRAFO_AXES.IN.2[4] = 0

$MC.TRAFO.GEOAX.ASSIGN.TAB.2[0] = 4
$MC.TRAFO.GEOAX.ASSIGN.TAB.2[1] = 0
$MC.TRAFO.GEOAX.ASSIGN.TAB.2[2] = 3

$MC.TRAANG.ANGLE.2 = -85.
$MC.TRAANG.PARALLEL.VELO.RES.2 = 0.2
$MC.TRAANG.PARALLEL.ACCEL.RES.2 = 0.2

$MC.TRAANG.BASE.TOOL.2[0] = 0.0
$MC.TRAANG.BASE.TOOL.2[1] = 0.0
$MC.TRAANG.BASE.TOOL.2[2] = 0.0

Part program:

; Frame settings
N820 $P.UIFR[1] = ctrans(x,1,y,2,z,3,b,4,c,5)
N850
N860 $P.CHBF0[0] = ctrans(x,10,y,20,z,30,b,40,c,15)
N870

; Tool selection, clamping compensation, plane selection
N890 T2 D1 G54 G17 G90 F5000 G64 SOFT
N900

; Approach start position
N920  G0 X20 Z10
N930
N940  if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,B,40,C,15)
N950  setal(61000)
N960  endif
N970  if $P_BFRAME <> $P_CHBFR[0]
N980  setal(61000)
N990  endif
N1000 if $P_IFRAME <> TRANS(X,1,Y,2,Z,3,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,C)
N1010 setal(61000)
N1020 endif
N1030 if $P_IFRAME <> $P_UIFR[1]
N1040 setal(61000)
N1050 endif
N1070 setal(61001)
N1080 endif
N1090
N1100 TRAANG(,1)
N1110
N1120 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,CAX,10,B,40,C,15)
N1130 setal(61000)
N1140 endif
N1150 if $P_BFRAME <> $P_CHBFR[0]
N1160 setal(61000)
N1170 endif
N1180 if $P_IFRAME <> CTRANS(X,1,Y,2,Z,3,CAX,1,B,4,C,5):CROT(X,10,Y,20,Z,30):CMIRROR(X,CAX,C)
N1190 setal(61000)
N1200 endif
N1210 if $P_IFRAME <> $P_UIFR[1]
N1220 setal(61000)
N1230 endif
N1250 setal(61001)
N1260 endif
N1270
N1280
N1290 $P_UIFR[1,x,tr] = 11
N1300 $P_UIFR[1,y,tr] = 14
N1310
N1320 g54
N1330

; Set frame
N1350 ROT RPL=-45
N1360 ATRANS X-2 Y10
N1370

; Four-edge roughing
N1390 G1 X10 Y-10 G41 OFFN=1; allowance 1 mm
N1400 X-10
N1410 Y10
N1420 X10
N1430 Y-10
N1440

; Tool change
N1460 G0 Z20 G40 OFFN=0
N1470 T3 D1 X15 Y-15
N1480 Z10 G41
N1490

; Four-edge finishing
N1510 G1 X10 Y-10
N1520 X-10
N1530 Y10
N1540 X10
N1550 Y-10
N1560

; Deselect frame
N1580 Z20 G40
N1590 TRANS
N1600
N1610 if $P_BFRAME <> CTRANS(X,10,Y,20,Z,30,CAX,10,B,40,C,15)
N1620 setal(61000)
N1630 endif
N1640 if $P_BFRAME <> $P_CHBFR[0]
N1650 setal(61000)
N1660 endif
10.5.6.3 Adapting active frames

The geometry axis configuration can change during program execution or on RESET. The number of available geometry axes can vary from zero to three. With unavailable geometry axes, components in the active frames (e.g., rotations) can lead to the active frames for this axis configuration becoming invalid. This is indicated by the alarms below:

Channel %1 block %2 rotation programmed for non-existent geometry axis

This alarm remains present until the frames have been changed accordingly.

The machine data below can be used to switch on the automatic adaptation of active frames, thus preventing alarm 16440:

MD24040 $MC_FRAME_ADAPT_MODE
Bitmask for adapting the active frames with reference to the axis constellation.

The following settings apply:

Bit 0: Rotations in active frames, which rotate coordinate axes with no geometry axes, are deleted from the active frames.
Bit 1: Shear angles in the active frames are orthogonalized.
Bit 2: Scalings of all geometry axes in the active frames are set to value 1.

With machine data setting
MD24040 $MC_FRAME_ADAPT_MODE = 1
all rotations in the active frames, which could produce coordinate axis movements for non-existent geometry axes, are deleted.
The data management frames are not changed. Only the possible rotations are applied when the data management frames are activated.

Example:
No y axis exists:
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3
$P_UIFR[1] = crot(x,45,y,45,z,45)

N390 G54 G0 X10 z10 f10000
if $P_IFRAME <> crot(y,45) ; Only the rotation about y is taken over.
    setal(61000)
endif

10.5.6.4 Mapped Frames

Overview

The "mapped frames" function supports the cross-channel consistent change of axial frames inside channel-specific or global data management frames. Using axial machine data, it is defined between which axes mapping is realized.

If frame mapping is, e.g. active for machine axes AX1 and AX4, and the axial frame of axis AX1 is changed in a channel-specific data management frame (e.g. basic frame $P_CHBFR[x]) (translation, fine translation, scaling, mirroring), then this frame data for AX1 and AX4 is transferred to all channel-specific data management frames (e.g. basic frame $P_CHBFR[x]) in all channels in which they are parameterized as channel axes.

Frame mapping is not realized when changing the axial frame data for the rotation.
Preconditions

The following requirements must be fulfilled for frame mapping:

- The data management frames used for mapping must be configured:
  MD28083 $MC_MM_SYSTEM_DATAFRAME_MASK (system frames)
- **Channel-specific** data management frames must be explicitly enabled for mapping:
  MD10616 $MN_MAPPED_FRAME_MASK (enable frame mapping)

*Note*
For **global** data management frames, mapping is always carried out. An enable is not required.

Parameterization

The parameterization of the mapping relationships is realized in the axis-specific machine data:

MD32075 $MA_MAPPED_FRAME[<AXn>] = "AXm"

AXn, AXm: Machine axis name with n, m = 1, 2, ... max. number of machine axes

Mapping rules

The following rules apply for frame mapping:

- The mapping is bidirectional.
  An axial frame can be written for AXn or AXm. The frame data is always accepted and taken for the other axis.

- All parameterized mapping relationships are always evaluated.
  When writing an axial frame of axes AXn, all mapping relationships are evaluated and the frame data accepted for all directly and indirectly involved axes.

- The mapping is global across all channels.
  When writing an axial frame of axis AXn or AXm for a channel-specific frame, the frame data is accepted for all channels in which AXn or AXm are parameterized as channel axes.

- When writing an axial frame using geometry or special axis name, the mapping relationships are evaluated via the machine axes currently assigned to the geometry or special axis.

- Mapping is frame-specific.
  When writing an axial frame, the frame data is only mapped within the same channel-specific or global data management frame.

*Note*

**Data consistency**

The user / machine manufacturer is solely responsible for ensuring that after a frame is written, consistent frame data is available in all channels, e.g. by using channel synchronization.
Figure 10-23 Mapping examples
Activating the data management frame

Data management frames can be written in the part program and via the user interface of SINUMERIK Operate. The following should be noted when activating the data management frames in the channels written directly and via frame mapping:

- **Writing in the part program**
  The data management frames must be explicitly activated in each channel (G500, G54 ... G599)

- **Writing via user interface**
  Data management frames are written via the user interface, e.g. by entering new zero offsets. A modified data management frame is immediately active in all of the involved channels if none of these channels is in the "Channel active" state. The data management frame is not active in any channel if one of the channels involved is in the "Channel active" state. The activation must then be explicitly programmed in each channel in the part program (G500, G54 ... G599). Or, the next time that the channel state changes, it becomes active after "Channel reset".

Example

The following channels and channel axes are parameterized at a control:

- **Channel 1**
  - Z: Geometry axis
  - AX1: Machine axis of geometry axis Z

- **Channel 2**
  - Z: Geometry axis
  - AX4: Machine axis of geometry axis Z

The zero point of the Z axis should always be the same in both channels:

- **Mapping relationship**: $MA\_MAPPED\_FRAME[AX1] = "AX4"

Programming in the part program

<table>
<thead>
<tr>
<th>Channel 1</th>
<th>Channel 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N100 WAIT (10,1,2)</td>
<td>N200 WAIT (10,1,2)</td>
</tr>
<tr>
<td>N120 WAIT (20,1,2)</td>
<td>N220 WAIT (20,1,2)</td>
</tr>
<tr>
<td>N130 G54</td>
<td>N230 G54</td>
</tr>
<tr>
<td>N140 IF ($P_IFRAME[0, Z, TR] &lt;&gt; 10)</td>
<td>N240 IF ($P_IFRAME[0, Z, TR] &lt;&gt; 10)</td>
</tr>
<tr>
<td>N150 SETAL(61000)</td>
<td>N250 SETAL(61000)</td>
</tr>
<tr>
<td>N160 ENDF</td>
<td>N260 ENDF</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Description:

N100 / N200  Channel synchronization for consistent writing and mapping of frame data
N110 Writing of the settable data management frame $P_{UIFR}[1]:
Moving the zero point of the Z axis to 10 mm
Mapping the axial frame data:
Channel 1: Z ≙ AX1 ⇔ channel 2: Z ≙ AX4
N120 / N220 Channel synchronization for consistent activation of new frame data
N130 / N230 Activating the new frame data
N140 / N240 Checking the zero point of the Z axis for: 10 mm

10.5.7 Predefined frame functions

10.5.7.1 Inverse frame

To round off the frame arithmetic, the part program provides a function which calculates the inverse frame from another frame. The chaining between a frame and its inverse frame always produces a zero frame.

FRAME INVFRAME( FRAME ) ⇒ zero frame

Frame inversion is an aid for coordinate transformations. Measuring frames are usually calculated in the WCS. If you should wish to transform this calculated frame into another coordinate system, i.e. the calculated frame should be entered into any desired frame within the frame chain, then the calculations below should be used.

The new complete frame is a chain of the old complete frame and the calculated frame.

$P_{ACTFRAME} = $P_{ACTFRAME} : $AC_{MEAS_FRAME}

The new frame in the frame chain is therefore:

Target frame is $P_{SETFRAME}:
$P_{SETFRAME} = $P_{ACTFRAME} : $AC_{MEAS_FRAME} : INVFRAME($P_{ACTFRAME}) :
$P_{SETFRAME}

The target frame is the nth channel basic frame $P_{CHBFRAME}[n]:

n = 0: TMP = $P_{PARTFRAME} : $P_{SETFRAME} : $P_{EXTFRAME} : $P_{NCBFRAME}[0..k]
n <> 0: TMP = $P_{PARTFRAME} : $P_{SETFRAME} : $P_{EXTFRAME} : $P_{NCBFRAME}[0..k] :
$P_{CHBFRAME}[0..n-1]
k = $MN_MM_NUM_GLOBAL_BASE_FRAMES
$P_{CHBFRAME}[n] = INVFRAME(TMP) : $P_{ACTFRAME} : $AC_{MEAS_FRAME} :
INVFRAME($P_{ACTFRAME}) : TMP : $P_{CHBFRAME}[n]

The target frame is $P_{IFRAME}:

TMP = $P_{PARTFRAME} : $P_{SETFRAME} : $P_{EXTFRAME} : $P_{BFRAME}
$P_{IFRAME} = INVFRAME(TMP) : $P_{ACTFRAME} : $AC_{MEAS_FRAME} :
INVFRAME($P_{ACTFRAME}) : TMP : $P_{IFRAME}

Application example:
A frame calculated, for example, via a measuring function, must be entered in the current SETFRAME such that the new complete frame is a chain of the old complete frame and the measurement frame. The SETFRAME is calculated accordingly by means of frame inversions.

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEF INT RETVAL</td>
<td></td>
</tr>
<tr>
<td>DEF FRAME TMP</td>
<td></td>
</tr>
<tr>
<td>$TC_DP1[1,1]=120</td>
<td>; Type</td>
</tr>
<tr>
<td>$TC_DP2[1,1]=-20.</td>
<td>; 0</td>
</tr>
<tr>
<td>$TC_DP3[1,1]= 10.</td>
<td>; (z) length correction vector</td>
</tr>
<tr>
<td>$TC_DP4[1,1]= 0.</td>
<td>; (y)</td>
</tr>
<tr>
<td>$TC_DP5[1,1]= 0.</td>
<td>; (x)</td>
</tr>
<tr>
<td>$TC_DP6[1,1]= 2.</td>
<td>; Radius</td>
</tr>
<tr>
<td>T1 D1</td>
<td></td>
</tr>
<tr>
<td>g0 x0 y0 z0 f10000</td>
<td></td>
</tr>
<tr>
<td>G54</td>
<td></td>
</tr>
<tr>
<td>$P_CHBFRAME[0] = CROT(Z,45)</td>
<td></td>
</tr>
<tr>
<td>$P_IFRAME[X,TR] = -SIN(45)</td>
<td></td>
</tr>
<tr>
<td>$P_IFRAME[Y,TR] = -SIN(45)</td>
<td></td>
</tr>
<tr>
<td>$P_PFRAME[Z,TR] = -45</td>
<td></td>
</tr>
<tr>
<td>$AC_MEAS_VALID = 0</td>
<td>; Measure corner with four measuring points</td>
</tr>
<tr>
<td>G1 X-1 Y-3</td>
<td>; 1. Approach measuring point</td>
</tr>
<tr>
<td>$AC_MEAS_LATCH[0] = 1</td>
<td>; 1. Store measuring point</td>
</tr>
<tr>
<td>G1 X5 Y-3</td>
<td>; 2. Approach measuring point</td>
</tr>
<tr>
<td>$AC_MEAS_LATCH[1] = 1</td>
<td>; 2. Store measuring point</td>
</tr>
<tr>
<td>G1 X-4 Y4</td>
<td>; 3. Approach measuring point</td>
</tr>
<tr>
<td>G1 X-4 Y1</td>
<td>; 4. Approach measuring point</td>
</tr>
<tr>
<td>$AC_MEAS_LATCH[3] = 1</td>
<td>; 4. Store measuring point</td>
</tr>
<tr>
<td>$AA_MEAS_SETPOINT[X] = 0</td>
<td>; Set position setpoint of the corner</td>
</tr>
<tr>
<td>$AA_MEAS_SETPOINT[Y] = 0</td>
<td></td>
</tr>
<tr>
<td>$AA_MEAS_SETPOINT[Z] = 0</td>
<td></td>
</tr>
<tr>
<td>$AC_MEAS_CORNER_SETANGLE = 90</td>
<td>; Define setpoint angle of intersection</td>
</tr>
<tr>
<td>$AC_MEAS_WP_SETANGLE = 30</td>
<td>; Measuring plane is G17</td>
</tr>
<tr>
<td>$AC_MEAS_ACT_PLANE = 0</td>
<td>; Select tool</td>
</tr>
<tr>
<td>$AC_MEAS_T_NUMBER = 1</td>
<td></td>
</tr>
<tr>
<td>$AC_MEAS_D_NUMBER = 1</td>
<td></td>
</tr>
<tr>
<td>$AC_MEAS_TYPE = 4</td>
<td>; Set measuring type on corner 1</td>
</tr>
<tr>
<td>RETVAL = MEASURE()</td>
<td>; Start measuring process</td>
</tr>
<tr>
<td>IF RETVAL &lt;&gt; 0</td>
<td></td>
</tr>
<tr>
<td>SETAL(61000 + RETVAL)</td>
<td></td>
</tr>
<tr>
<td>ENDF</td>
<td></td>
</tr>
<tr>
<td>IF $AC_MEAS_WP_ANGLE &lt;&gt; 30</td>
<td></td>
</tr>
<tr>
<td>SETAL(61000 + $AC_MEAS_WP_ANGLE)</td>
<td></td>
</tr>
<tr>
<td>ENDF</td>
<td></td>
</tr>
</tbody>
</table>
**Program code**

IF $AC_MEAS_CORNER_ANGLE <> 90
   SETAL(61000 + $AC_MEAS_CORNER_ANGLE)
ENDIF

; Transform measured frame and write to $P_SETFRAME so that a complete frame is created,
; which linked from the old total frame results in the measurement frame.
$P_SETFRAME = $P_ACTFRAME : $AC_MEAS_FRAME : INVFRAME($P_ACTFRAME) : $P_SETFRAME

$P_SETFR = $P_SETFRAME

G1 X10

G1 X0

Y10

X0

Y0

M30

---

**10.5.7.2 Additive frame in frame chain**

Measurements on the workpiece or calculations in the part program and cycles generally produce a frame that is applied additively to the current complete frame. The WCS and thus the programming zero must, therefore, be displaced and possibly rotated. This measured frame is available as a temporary frame and not yet actively included in the frame chain. This function is used to calculate and possible activate this frame:

```
INT ADDFRAME(FRAME,STRING)
```

**Programming**

Parameter 1: Type: FRAME  Additively measured or calculated frame
### Parameter 2:

**Type:** STRING

Strings for current frames:
- "$P_CYCFRAME"
- "$P_ISO4FRAME"
- "$P_PFRAME"
- "$P_WPFRAME"
- "$P_TOOLFRAME"
- "$P_IFRAME"
- "$P_CHBFRAME[0..16]"
- "$P_NCBFRAME[0..16]"
- "$P_ISO1FRAME"
- "$P_ISO2FRAME"
- "$P_ISO3FRAME"
- "$P_EXTFRAME"
- "$P_SETFRAME"
- "$P_PARTFRAME"

Strings for data management frames:
- "$P_CYCFR"
- "$P_ISO4FR"
- "$P_TRAFR"
- "$P_WPFR"
- "$P_TOOLFR"
- "$P_UIFR[0..99]"
- "$P_CHBFR[0..16]"
- "$P_NCBFR[0..16]"
- "$P_ISO1FR"
- "$P_ISO2FR"
- "$P_ISO3FR"
- "$P_EXTFR"
- "$P_SETFR"
- "$P_PARTFR"

### Function value:

**Type:** INT

<table>
<thead>
<tr>
<th>Value</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
</tr>
<tr>
<td>1</td>
<td>Specified target (string) is wrong</td>
</tr>
<tr>
<td>2</td>
<td>Target frame is not configured</td>
</tr>
<tr>
<td>3</td>
<td>Rotation in frame is not permitted</td>
</tr>
</tbody>
</table>

The `ADDFRAME()` function calculates the target frame, which is specified by the STRING. The target frame is calculated in such a way that the new complete frame appears as a chain of the old complete frame and the transferred frame, e.g.:

\[
\text{ERG} = \text{ADDFRAME}(\text{TMPFRAME}, "$P\_SETFRAME")
\]

The new complete frame is calculated to be:

\[
$P\_ACTFRAME_{\text{new}} = $P\_ACTFRAME_{\text{old}} : \text{TMPFRAME}
\]

If a current frame has been specified as a target frame, then the new complete frame becomes active at the preprocessing stage. If the target frame is a data management frame, then the frame is not operative until it is explicitly activated in the part program.

The function does not set any alarms, but returns the error codes via the return value. The cycle can react according to the error codes.

### 10.5.8 Functions

#### 10.5.8.1 Setting zeros, workpiece measuring and tool measuring

PRESET is achieved using HMI operator actions or measuring cycles. The calculated frame can be written to system frame `SETFRAME`. The position setpoint of an axis in the WCS can be altered when the actual-value memory is preset.
"Scratching" means workpiece and tool measuring. The position of the workpiece in relation to an edge, a corner or a hole can be measured. To determine the zero position of the workpiece or the hole, position setpoints can be added to the measured positions in the WCS. The resultant offsets can be entered in a selected frame. In tool measuring, the length or radius of a tool can be measured using a measured reference part.

Measurements can be taken via operator actions or measuring cycles. Communication with the NCK takes place via predefined system variables. In the NCK, the calculation is made by using a HMI operator action to activate a PI service, or by using a part-program command from the measuring cycles. A tool and a plane can be selected as a basis for the calculation. The calculated frame is entered in the result frame.

10.5.8.2 Zero offset external via system frames

Zero offset via PLC or part program

The amount of the external zero offset can be specified through HMI and PLC via BTSS or programmed in the part program via the axis-specific system variable $AA_ETRANS [<Axis>] = <Value>.

Activation
The activation of the external zero offset takes place through the interface signal: DB31, ..., DBX3.0 (accept external zero offset)

Behavior
After the activation the respective specified axis-specific amount of the external zero offset is traversed outside for each axis with the next possible traversing block.

Next possible means that enough dynamic reserves must be available for the respective axis for traversing the zero offset. If the axis is traversed in the next traversing block after activation owing to its programming with the maximum speed, dynamic reserve is no longer available for traversing the external zero offset.

Together with the continuous path mode G64 the traversing of the offset can stretch over several part program blocks.

Zero offset via system frame

The external zero offset can then also be managed and activated by the functionality described above via the system frame $P_EXTFRAME.

configuring
The configuring of the external zero offset is done via the system frame $P_EXTFRAME via bit1 = 1 in the machine data: MD28082 $MC_MM_SYSTEM_FRAME_MASK = 'B0010'
The amount for the external zero offset can be specified manually via the HMI user interface and the PLC user program via BTSS or programmed in the part program via the axial system variable $AA_ETRANS[<Axis>].

Activation
The activation of the external zero offset takes place through the interface signal: DB31, ..., DBX3.0 (accept external zero offset)

Behavior
Upon activation of the external zero offset the traversing movements of all axes, except
command and PLC axes, are stopped immediately and the advance is reorganized. The rough offset of the current system frame and of the system frame in data management is set to the value of the axial system variable $AA\_ETRANS[<axis>]$. Thereafter, the offset is traversed first and then the interrupted movement is continued.

**Behavior for incremental dimension**

In case of active incremental dimension G91 and machine data:

MD42440 $MC\_FRAME\_OFFSET\_INCR\_PROG$ (zero offset in frames) = 0

traversing the offset is done in the scope of the external zero offset via system frame, despite opposite configuring of the machine data, with the approach block, although it is specified by a frame.

**Note**

The external zero offset always acts absolutely.

---

**10.5.8.3 Toolholder**

**Translations**

With kinematics of type "P" and "M", the selection of a toolholder activates an additive frame (table offset of the orientable toolholder) which takes into account the work offset as a result of the rotation of the table. The work offset is entered in the system frame $P\_PARTFR$. In this case the translatory component of this frame is overwritten. The other frame components are retained.

The system frame $P\_PARTFR$ must be enabled via the following machine data:

MD28082 $MC\_MM\_SYSTEM\_FRAME\_MASK$, bit 2 = 1 (system frame for TCARR and PAROT)

**Note**

Alternatively, the offset can also parameterized via machine data to record the table offset:

MD20184 $MC\_TOCARR\_BASE\_FRAME\_NUMBER$ = <number of the basic frame>

This option is only for compatibility reasons to older software versions. You are strongly recommended **not** to use this procedure any longer.

A frame offset as a result of a toolholder change becomes effective immediately on selection of TCARR=... A change in the tool length, on the other hand, only becomes effective immediately if a tool is active.

A frame rotation is not executed with the activation, or a rotation which is already active is not changed. As in case T (only the tool can be rotated), the position of the rotary axes used for the calculation is dependent on the command TCOFR/TCOABS and determined from the rotation component of an active frame or from the $TC\_CARR$ entries. Activation of a frame changes the position in the workpiece coordinate system accordingly, without compensating motion by the machine itself.

The ratios are shown in the figure below:
With kinematics of type M (tool and table are each rotary around one axis), the activation of a toolholder with TCARR simultaneously produces a corresponding change in the effective tool length (if a tool is active) and the work offset.

**Rotations**

Depending on the machining task, it is necessary to take into account not only a work offset (whether as frame or as tool length) when using a rotary toolholder or table, but also a rotation. However, the activation of an orientable toolholder never leads directly to a rotation of the coordinate system.

If only the tool can be rotated, a frame can be defined for it using TOFRAME or TOROT.

With rotary tables (kinematics types P and M), activation with TCARR similarly does not lead to an immediate rotation of the coordinate system, i.e. even though the zero point of the coordinate system is offset relative to the machine, while remaining fixed relative to the zero point of the workpiece, the orientation remains unchanged in space.

If the coordinate system needs to be fixed relative to the workpiece, i.e. not only offset relative to the original position but also rotated according to the rotation of the table, then PAROT can be used to activate such a rotation in a similar manner to the situation with a rotary tool.

With PAROT, the translations, scalings and mirroring in the active frame are retained, but the rotation component is rotated by the rotation component of an orientable toolholder corresponding to the table. The entire programmable frame including its rotation component remains unchanged.

The rotation component that describes the rotation of the table is then either entered in the system frame $PARTFR or in the basic frame parameterized with MD20184 $MC_TOCARR_BASE_FRAME_NUMBER:

$MC_MM_SYSTEM_FRAME_MASK, bit 2 = <value>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rotation component → $PARTFR</td>
</tr>
<tr>
<td>0</td>
<td>Rotation component → MD20184 $MC_TOCARR_BASE_FRAME_NUMBER</td>
</tr>
</tbody>
</table>

As with the note made in the description of the table offset, the second alternative here is not recommended for use with new systems.
The rotation component of the part frame can be deleted with PAROTOF, irrespective of whether this frame is in a basic or a system frame. The translation component is deleted when a toolholder which does not produce an offset is activated or a possibly active orientable toolholder is deselected with TCARR=0.

PAROT or TOROT take into account the overall orientation change in cases where the table or the tool are oriented with two rotary axes. With mixed kinematics, only the corresponding component caused by a rotary axis is considered. It is thus possible, for example, when using TOROT, to rotate a workpiece such that an inclined plane lies parallel to the XY plane fixed in space, whereby rotation of the tool must be taken into account in machining where any holes to be drilled, for example, are not perpendicular to this plane.

Example
On a machine, the rotary axis of the table points in the positive Y direction. The table is rotated by +45 degrees. PAROT defines a frame which similarly describes a rotation of 45 degrees around the Y axis. The coordinate system is not rotated relative to the actual environment (marked in the figure with “Position of the coordinate system after TCARR”), but is rotated by -45 degrees relative to the defined coordinate system (position after PAROT). If this coordinate system is defined with ROT Y-45, for example, and if the toolholder is then selected with active TCOFR, an angle of +45 degrees will be determined for the rotary axis of the toolholder.

Language command PAROT is not rejected if no orientable toolholder is active. However, such a call then produces no frame changes.

Machining in direction of tool orientation
Particularly on machines with tools that can be oriented, traversing should take place in the tool direction (typically, when drilling) without activating a frame (e.g. using TOFRAME or TOROT), on which one of the axes points in the direction of the tool. This is also a problem if, when carrying out inclined machining operations, a frame defining the inclined plane is active, but the tool cannot be set exactly perpendicularly because an indexed toolholder (Hirth tooth system) prevents free setting of the tool orientation. In these cases it is then necessary - contrary to the motion actually requested perpendicular to the plane - to drill in the tool direction, as the drill would otherwise not be guided in the direction of its longitudinal axis (tool breaks).

The end point of such a motion is programmed with MOVT= .... The programmed value is effective incrementally in the tool direction as standard. The positive direction is defined from the tool tip to the tool adapter. The content of MOVT is thus generally negative for the infeed motion (when drilling), and positive for the retraction motion. This corresponds to the situation with normal paraxial machining, e.g. with G91 Z ....

Instead of MOVT= .... it is also possible to write MOVT=IC( .... ) if it is to be plainly visible that MOVT is to function incrementally. There is no functional difference between the two forms.

If the motion is programmed in the form MOVT=AC( .... ), MOVT functions absolutely. In this case a plane is defined which runs through the current zero point, and whose surface normal vector is parallel to the tool orientation. MOVT then gives the position relative to this plane (see figure). The reference plane is only used to calculate the end position. Active frames are not affected by this internal calculation.
Programming with MOVT is independent of the existence of a toolholder that can be oriented. The direction of the motion is dependent on the active plane. It runs in the directions of the vertical axes, i.e. with G17 in Z direction, with G18 in Y direction and with G19 in X direction. This applies when no orientable toolholder is active and for the case of an orientable toolholder without rotary tool or with a rotary tool in its basic setting.

MOVT acts similarly for active orientation transformation (3-, 4-, 5-axis transformation).

If in a block with MOVT the tool orientation is changed simultaneously (e.g. active 5-axis transformation by means of simultaneous interpolation of the rotary axes), the orientation at the start of the block is decisive for the direction of motion of MOVT. With an active 5-axis transformation, the path of the tool center point (TCP) is not affected by the change of orientation, i.e. the path remains a straight line and its direction is determined by the tool orientation at the start of the block.

If MOVT is programmed, linear or spline interpolation must be active (G0, G1, ASPLINE, BSPLINE, CSPLINE). Otherwise, an alarm is produced. If a spline interpolation is active, the resultant path is generally not a straight line, since the end point calculated by MOVT is treated as if it had been programmed explicitly with X, Y, Z.

A block with MOVT must not contain any programming of geometry axes (alarm 14157).

Definition of frame rotations with solid angles

Where a frame is to be defined to describe a rotation around more than one axis, this is achieved through chaining individual rotations. The next rotation is always performed in the currently rotated coordinate system. This applies both to programming in one block, e.g. with ROT X..., Y..., Z..., and when constructing a frame in several blocks, e.g. in the form:

N10 ROT Y...
N20 AROT X...
N30 AROT Z...

In workpiece drawings, inclined surfaces are frequently described by way of solid angles, i.e. the angles which the intersection lines of the inclined plane form with the main planes (X-Y, Y-
Z, Z-X planes) (see figure). The machine operator is not expected to convert these solid angles into the angles of rotation of a chaining of individual rotations.

For this reason, the language commands ROTS, AROTS and CROTS are used, with which the rotations can be immediately described as solid angles.

The orientation of a plane in space is defined unambiguously by specifying two solid angles. The third solid angle is derived from the first two. Therefore, a maximum of two solid angles may be programmed, e.g. in the form ROTS X10 Y15. If a third solid angle is specified, an alarm will be triggered.

It is permissible to specify a single solid angle. The rotations which are performed with ROTS or AROTS in this case are identical to those for ROT and AROT.

An expansion of the existing functionality arises only in cases where exactly two solid angles are programmed.

The two programmed axes define a plane, the non-programmed axis defines the related third axis of a right-hand coordinate system. Which axis is first and which second is then unambiguously defined for both programmed axes (the definition corresponds to those found in the plane definition of G17/G18/G19). The angle programmed with the axis letter of an axis
of the plane then specifies the axis, around which the other axis of the plane must be rotated in order to move this into the line of intersection, which the rotated plane forms with the plane surrounded by the other and the third axis. This definition ensures that, in the case that one of the two programmed angles is towards zero, the defined plane enters the plane, which is created if only one axis is programmed (e.g. with ROT or AROT).

The diagram shows an example where X and Y are programmed. Y specifies the angle here, through which the X axis must rotate around the Y axis to bring the X axis to the line of intersection formed by the inclined plane and the X-Z plane. The same principle applies for the programmed value of X.

Note

In the shown position of the inclined plane the value of Y is positive, that of X on the other hand negative.

The specification of the solid angle does not define the orientation of the two-dimensional coordinate system within the plane (i.e. the angle of rotation around the surface normal vector). The position of the coordinate system is thus determined so that the rotated first axis lies in the plane which is formed by the first and third axes of the non-rotated coordinate system.

This means that

- When programming X and Y the new X axis lies in the old Z-X plane.
- When programming Z and X the new Z axis lies in the old Y-Z plane.
- When programming Y and Z the new Y axis lies in the old X-Y plane.

If the required coordinate system does not correspond to this basic setting, then an additional rotation must be performed with AROT...

The programmed solid angles are converted to the equivalent Euler angles according to the zy'x" convention (RPY angle) or zx'z" convention when entering depending on the machine data:

MD10600 $MN_FRAME_ANGLE_INPUT_MODE.
These then appear in the display.

Frame rotation in tool direction

With the language command TOFRAME, which also existed in older software versions, it is possible to define a frame whose Z axis points in the tool direction. An already programmed frame is then overwritten by a frame which describes a pure rotation. Any work offsets, mirrorings or scalings existing in the previously active frame are deleted. This response is sometimes interfering. It is often particularly practical to retain a work offset, with which the reference point in the workpiece is defined.

The language command TOROT is then also used. This command overwrites only the rotation component in the programmed frame and leaves the remaining components unchanged. The rotation defined with TOROT is the same as that defined with TOFRAME.

TOROT is, like TOFRAME, independent of the availability of an orientable toolholder. This language command is also especially useful for 5-axis transformations.

The new language command TOROT ensures consistent programming with active orientable toolholders for each kinematics type.
TOFRAME or TOROT defines frames whose Z direction points in the tool direction. This definition is suitable for milling, where G17 is usually active. However, particularly with turning or, more generally, when G18 or G19 is active, it is desirable that frames which will be aligned on the X or Y axis, can be defined. For this purpose, the following commands are available in G group 53:

- TOFRAMEX, TOFRAMEY, TOFRAMEZ
- TOROTX, TOROTY, TOROTZ

The appropriate frames can be defined with these commands. The functions of TOFRAME and TOFRAMEZ or of TOROT and TOROTZ are identical to one another.

The frames resulting from TOROT or TOFRAME can be written in a separate system frame ($P\_TOOLFR). The programmable frame is then retained unchanged.

- Requirement: MD28082 $MC\_MM\_SYSTEM\_FRAME\_MASK, bit 3 = 1

When programming TOROT or TOFRAME, etc. response is identical, with or without a system frame. Differences occur when the programmable frame is processed further elsewhere.

**Note**

In new systems, it is recommended that only the intended system frame is used for frames produced by the commands of G group 53.

**Example**

TRANS is programmed after TOROT. TRANS without specified parameters deletes the programmable frame. In the variant without a system frame, this also deletes the frame component of the programmable frame produced by TOROT, but if the TOROT component is in the system frame, it is retained.

TOROT or TOFRAME, etc. are disabled with language command TOROTOF. TOROTOF deletes the entire system frame $P\_TOOLFR. If the programmable frame (old variant) and not the system frame is described by commands TOFRAME, etc. TOROT only deletes the rotation component and leaves the remaining frame components unchanged.

If a rotating frame is already active before language command TOFRAME or TOROT is activated, a request is often made that the newly defined frame should deviate as little as possible from the old frame. This is the case, for example, if a frame definition needs to be modified slightly because the tool orientation cannot be set freely on account of Hirth-toothed rotary axes. The language commands uniquely define the Z direction of the new frame.

**Frame definition for TOFRAME, TOROT and PAROT**

The following machine data can be used to select one of two variants for the position of the X axis and Y axis. However, in both cases there is no reference to the previously active frame.

MD21110 $MC\_X\_AXIS\_IN\_OLD\_X\_Z\_PLANE (coordinate system for automatic frame definition)

For this reason, it is recommended that the following setting data be used instead so that the behavior of TOFRAME and TOROT can be specifically controlled.
SD42980 $SC_TOFRAME_MODE (frame definition for TOFRAME, TOROT and PAROT)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The new X direction is chosen to lie in the X-Z plane of the old coordinate system. In this setting, the angle difference between the old and new Y axis will be minimal.</td>
</tr>
<tr>
<td>2</td>
<td>The new Y direction is chosen to lie in the Y-Z plane of the old coordinate system. In this setting the angle difference between the old and new X axis will be minimal.</td>
</tr>
<tr>
<td>3</td>
<td>The value chosen is the mean value of the two settings, which would have been chosen with 1 and 2.</td>
</tr>
</tbody>
</table>

A detailed description of all parameterization options can be found in:

References
Detailed Machine Data Description

Special features and extensions

The behavior for value = 1 and 2 is achieved when starting from an arbitrary position of the X and Y axes, by rotating the coordinate system around the Z axis until the desired setting is reached.

The behavior for value = 3 is achieved by executing a rotation whose value is the exact mean of these two angles. However, this only applies for the case that the old and new Z direction enclose an angle of less than 90 degrees.

With value = 1, both the old and new X axes form an angle of under 90 degrees, with value = 2 the same is true of the Y axis (the relevant axes point in "approximately" the same direction). If the two Z directions form an angle of more than 90 degrees, however, the conditions of an angle < 90 degrees between the old and new axes can no longer be met simultaneously for both X and Y. In this case, priority is given to the X direction, i.e. a mean value is taken from the direction for 1 and the negative direction for 2.

If one of the commands TOFRAMEX, TOFRAMEY, TOROTX, TOROTY is programmed in place of TOFRAME(Z) or TOROT(Z), the descriptions for adapting the axis directions perpendicular to the main direction are also valid for the cyclically exchanged axes. The assignments in the table below are then valid:

<table>
<thead>
<tr>
<th>Tool direction (vertical axis)</th>
<th>TOFRAME, TOFRAMEZ</th>
<th>TOROT, TOROTZ</th>
<th>TOFRAMEY</th>
<th>TOROTY</th>
<th>TOFRAMEX</th>
<th>TOROTX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secondary axis (horizontal axis)</td>
<td>Z</td>
<td>X</td>
<td>Y</td>
<td></td>
<td>Z</td>
<td>Y</td>
</tr>
<tr>
<td>Secondary axis (ordinate)</td>
<td>X</td>
<td>Y</td>
<td>X</td>
<td></td>
<td>Z</td>
<td></td>
</tr>
</tbody>
</table>

Example

N90  $SC_TOFRAME_MODE=1
N100  ROT Z45
N110  TCARR=1 TCOABS T1 D1
N120  TOROT
N100 describes a rotation by 45 degrees in the X-Y plane. It is assumed that the toolholder activated in N110 rotates the tool by 30 degrees around the X axis, i.e. the tool lies in the Y-Z plane and is rotated by 30 degrees relative to the Z axis. As a result the Z axis of the frame newly defined in N120 also points in this direction (independently of the value in setting data SD42980 $SC_TOFRAME_MODE in N90).

SD42980 $SC_TOFRAME_MODE == 1:

![Diagram showing rotation](attachment:rotation_diagram.png)

The old and new X and X' coordinate axes coincide in the projection in the direction of the old Z axis. The old and new Y and Y' coordinate axes form an angle of 8.13 degrees (right angles are generally not retained in the projection).

SD42980 $SC_TOFRAME_MODE == 2:
Y and Y' coincide and X and X' form an angle of 8.13 degrees.

SD42980 $SC_TOFRAME_MODE == 3:
X and X' as well as Y and Y' each form an angle of 4.11 degrees.

Note
The named angles (8.13 and 4.11 degrees) are the angles, which the projections of the axes form in the X-Y plane. They are not the spatial angles of these axes.

**TCARR (request toolholder) and PAROT (align workpiece coordinate system on the workpiece)**

**TCARR** uses the basic frame identified by following machine data:
MD20184 $MC_TOCARR_BASE_FRAME_NUMBER.

A system frame can be created for **TCARR** and **PAROT** alone, in order to avoid conflicts with systems, which already use all the basic frames.

**PAROT**, **TOROT** and **TOFRAME** have previously changed the rotation component of the programmable frames. In this case, it is not possible to shut down **PAROT** or **TOROT** separately. On **RESET**, the programmable frame is deleted, which means that after changing the operating mode to **JOG**, the rotation component of **PAROT** and **TOROT** is no longer available. The user must also have unrestricted access to the programmable frame. Frames produced by **PAROT** and **TOROT** must be able to be archived and reloaded via data backup.
The system frame for **TCARR** and **PAROT** is configured with:

MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit 2 = 1

The following machine data is then no longer evaluated:

MD20184 $MC_TOCARR_BASE_FRAME_NUMBER

If the system frame for **TCARR** is configured, **TCARR** and **PAROT** describe that corresponding system frame; otherwise the basic frame identified by machine data MD20184 is described.

With kinematics systems of the types P and M, **TCARR** will enter the table offset of the orientable toolholder (work offset resulting from the rotation of the table) as a translation into the system frame. **PAROT** converts the system frame so that a workpiece-related WCS results.

The system frames are stored retentively and therefore retained after a reset. The system frames also remain active in the case of a mode change.

For the display, the commands **PAROT** and **TOROT**, **TOFRAME** are each assigned to a separate G code group.

**PAROTOF**

**PAROTOF** is the switch off command for **PAROT**. This command deletes the rotations in the system frame for **PAROT**. In so doing, the rotations in the current $P_PARTFRAME and in the data management frame $P_PARTFR are deleted. The position of the coordinate system is then recreated according to **TCARR**. **PAROTOF** is in the same G group as **PAROT** and appears therefore in the G code display.

**TOROT (align Z axis of the WCS by rotating the frame parallel to the tool orientation) and TOFRAME (ditto.)**

The system frame for **TOROT** and **TOFRAME** is activated via the following machine data:

MD28082 $MC_MM_SYSTEM_FRAME_MASK, bit 3 = 1

This system frame is located before the programmable frame in the frame chain. The Szs coordinate system is located before the programmable frame.

**TOROTOF**

**TOROTOF** is the switch off command for **TOROT** and **TOFRAME**. This command deletes the corresponding system frame. The current $P_TOOLFRAME and the data management frame $P_TOOLFR are also deleted. **TOROTOF** is in the same G group as **TOROT** and **TOFRAME** and appears therefore in the G code display.

**Example**

Use of an orientable toolholder with resolved kinematics.

```
N10 $TC_DP1[1,1]=120
N20 $TC_DP3[1,1]= 13 ; Tool length 13 mm
 ; Definition of toolholder 1:
N30 $TC_CARR1[1] = 0 ; X component of 1st offset vector
N40 $TC_CARR2[1] = 0 ; Y component of 1st offset vector
```
10.5 Frames

10.5.9 Subprograms with SAVE attribute (SAVE)

For various frames, the behavior regarding subprograms can be set using the SAVE attribute.
Settable frames G54 to G599

The behavior of the adjustable frames can be set using MD10617
$MN_FRAME_SAVE_MASK.BIT0:

- BIT0 = 0
  Using the subprogram, if only the values of the active adjustable frame are changed using the system variable $P_IFRAME, but the G functions are kept, then the change is also kept after the end of the subprogram.
- BIT0 = 1
  With the end of the subprogram, the adjustable frame, G function and values, active before the subprogram call, are reactivated.

Basic frames $P_CHBFR[ ] and $P_NCBFR[ ]

The behavior of the basic frame can be set using MD10617
$MN_FRAME_SAVE_MASK.BIT1:

- BIT1 = 0
  If the active basic frame is changed by the subprogram, the change remains effective even after the end of the subprogram.
- BIT1 = 1
  With the end of the subprogram the basic frame which is active before the subroutine call is reactivated.

Programmable frame

With the end of the subprogram the programmable frame active before the subroutine call is reactivated.

System frames

If the system frames are changed by the subprogram, the change remains effective even after the end of the subprogram.

10.5.10 Data backup

Data block _N_CHANx_UFR is used to archive the system frames.

Machine data
MD28082 $MC_MM_SYSTEM_FRAME_MASK
should not have changed between saving and reintroducing the saved system frames. If it has changed then it is possible that saved system frames could no longer be loaded. In this case, the loading process triggers an alarm.

Data backup always takes place in accordance with the currently valid geometry axis assignment, not in accordance with the axis configurations set in the machine data.
The machine data $MC_{\text{MM\_SYSTEM\_DATAFRAME\_MASK}}$
can be used to configure data management frames for the system frames.
If you do not want a data management frame for a system frame, the frame does not have to
be saved. With G500, G54 to G599, the active frame is retained.

A separate data block _N_NC_UFR is used to archive global frames.
The block requested by the HMI is created if the machine data
MD18601 $\text{MN\_MM\_NUM\_GLOBAL\_USER\_FRAMES}$
or
MD18602 $\text{MN\_MM\_NUM\_GLOBAL\_BASE\_FRAMES}$
has a value greater than zero.
Channel-specific frames are saved in data block _N_CHANx_UFR.

In certain circumstances, alarms could be triggered when reintroducing saved data, if the frame
affiliates, be they NCU global or channel-specific, have been changed using machine data.

Data backup always takes place in accordance with the axis configuration set in the machine
data, not in accordance with the currently valid geometry axis assignment.

### 10.5.11 Positions in the coordinate system

The setpoint positions in the coordinate system can be read via the following system variables.
The actual values can be displayed in the WCS, SZS, BZS or MCS via the PLC. There is a
softkey for actual-value display in MCS/WCS. Machine manufacturers can define on the PLC side, which coordinate system corresponds to the workpiece coordinate system on their
machines. The HMI requests the appropriate actual values from the NCK.

$AA_{\text{IM[axis]}}$
The setpoints in the machine coordinate system can be read for each axis using the variables $AA_{\text{IM[axis]}}$.

$AA_{\text{IEN[axis]}}$
The setpoints in the settable zero system (SZS) can be read for each axis using the variables $AA_{\text{IEN[axis]}}$.

$AA_{\text{IBN[axis]}}$
The setpoints in the basic zero system (BZS) can be read using the variable $AA_{\text{IBN[axis]}}$.

$AA_{\text{IW[axis]}}$
The setpoints in the workpiece coordinate system (WCS) can be read using the variable $AA_{\text{IW[axis]}}$. 
10.5.12 Control system response

10.5.12.1 POWER ON

<table>
<thead>
<tr>
<th>Frame</th>
<th>Frame conditions after POWER ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmable frame: $P_FRAME</td>
<td>Deleted.</td>
</tr>
</tbody>
</table>
| Settable frames: $P_IFRAME | Are retained, depending on:  
  - MD24080 $MC_USER_FRAME_POWERON_MASK, bit 0  
  - MD20152 $MC_GCODE_RESET_MODE[7] |
| Grinding frames: $P_GFRAMES | Are retained, depending on:  
  - MD24080 $MC_USER_FRAME_POWERON_MASK, bit 0  
  - MD20152 $MC_GCODE_RESET_MODE[7] |
| Complete basic frame: $P_ACTBF-RAME | Is retained, depending on:  
  - MD20110 $MC_RESET_MODE_MASK bit 0 and bit 14  
  Basic frames can be deleted via machine data:  
  - MD10615 $MN_NCBFRAME_POWERON_MASK  
  - MD24004 $MC_CHBFFRAME_POWERON_MASK |
| System frames: $P_PARTFRAME, $P_SET-FRAME, $P_ISO1FRAME, $P_ISO2FRAME, $P_ISO3FRAME, $P_TOOL-FRAME, $P_WPFRAME, $P_TREFRAME, $P_ISO4FRAME, $P_RELFRA, $P_CACFRAME | Retained  
Individual system frames can be deleted via machine data:  
- MD24008 $MC_CHSFFRAME_POWERON_MASK  
Deletion of system frame is executed in the data management on first priority. |
| External work offset: $P_EX-TFRAME | Is retained, but has to be activated again. |
| DRF offset | Deleted. |

10.5.12.2 Mode change

System frames
The system frames are retained and remain active.

JOG mode
In JOG mode, the frame components of the current frame are only taken into account for the geometry axes if a rotation is active. No other axis-specific frames are taken into account.

PLC and command axes
The response for PLC and command axes can be set via machine data:

MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED (frame or HL correction is not permissible)
10.5.12.3  RESET, end of part program

RESET responses of basic frames

The RESET response of basic frames is set via the machine data:

MD20110 $MC_RESET_MODE_MASK (definition of initial control settings after RESET/TP-End)

RESET responses of system frames

The system frames are retained in the data management after a Reset.

The machine data below can be used to configure the activation of individual system frames:

MD24006 $MC_CHSFRAME_RESET_MASK (active system frames after RESET)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>System frame for actual value setting and scratching is active after RESET.</td>
</tr>
<tr>
<td>1</td>
<td>System frame for external work offset is active after RESET.</td>
</tr>
<tr>
<td>2</td>
<td>Is not evaluated.</td>
</tr>
<tr>
<td>3</td>
<td>Is not evaluated.</td>
</tr>
<tr>
<td>4</td>
<td>System frame for workpiece reference point is active after RESET.</td>
</tr>
<tr>
<td>5</td>
<td>System frame for cycles is active after RESET.</td>
</tr>
<tr>
<td>6</td>
<td>Reserved, RESET response depends on MD20110 $MC_RESET_MODE_MASK.</td>
</tr>
<tr>
<td>7</td>
<td>System frame $P_ISO1FR is active after RESET.</td>
</tr>
<tr>
<td>8</td>
<td>System frame $P_ISO2FR is active after RESET.</td>
</tr>
<tr>
<td>9</td>
<td>System frame $P_ISO3FR is active after RESET.</td>
</tr>
<tr>
<td>10</td>
<td>System frame $P_ISO4FR is active after RESET.</td>
</tr>
<tr>
<td>11</td>
<td>System frame $P_RELFR is active after RESET.</td>
</tr>
</tbody>
</table>

RESET response of the system frames of TCARR, PAROT, TOROT and TOFRAME

The RESET response of the system frames of TCARR, PAROT, TOROT and TOFRAME depends on the G code RESET setting.
The setting is made with machine data:

- **MD20110 $MC_RESET_MODE_MASK, bit\(n\)** (definition of initial control setting after RESET / TP end)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>TCARR and PAROT system frames are retained as before the RESET.</td>
</tr>
</tbody>
</table>
| 0   | 1     | MD20152 $MC_GCODE_RESET_MODE[51] = 0  
- MD20150 $MC_GCODE_RESET_VALUES[51] = 1  PAROTOF  
- MD20150 $MC_GCODE_RESET_VALUES[51] = 2  PAROT |
|     |       | MD20152 $MC_GCODE_RESET_MODE[51] = 1  TCARR and PAROT are retained. |
|     |       | MD20152 $MC_GCODE_RESET_MODE[52] = 0  
- MD20150 $MC_GCODE_RESET_VALUES[52] = 1  TOROTOF  
- MD20150 $MC_GCODE_RESET_VALUES[52] = 2  TOROT  
- MD20150 $MC_GCODE_RESET_VALUES[52] = 3  TOFRAME |
|     |       | MD20152 $MC_GCODE_RESET_MODE[52] = 1  TOROT and TOFRAME are retained. |

Description of the further bits for **bit 0 == 1**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>0</td>
<td>Chained complete basic frame is deleted.</td>
</tr>
</tbody>
</table>
| 14  | 1     | The complete basic frame results from the following additional machine data:  
MD24002 $MC_CHBFRAME_RESET_MASK, bit\(n\) = 1  
n: nth channel-specific basic frame is calculated into the chained complete basic frame.  
MD10613 $MN_NCBFRAME_RESET_MASK, bit\(n\) = 1  
n: nth NCU-global basic frame is calculated into the chained complete basic frame. |

**TCARR** and **PAROT** are two independent functions, which describe the same frame. With **PAROTOF**, the component of **TCARR** is not activated on RESET.

- **MD20152 $MC_GCODE_RESET_MODE[]** (RESET response of G groups)  
- **MD20150 $MC_GCODE_RESET_VALUES** (RESET position of G groups)

## Frame conditions after RESET / part program end

<table>
<thead>
<tr>
<th>Frame</th>
<th>Condition after RESET / part program end</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmable frame: $P_PFRA ME</td>
<td>Deleted.</td>
</tr>
</tbody>
</table>
| Settable frames: $P_IFRAME | Are retained, depending on:  
- MD20110 $MC_RESET_MODE_MASK  
- MD20152 $MC_GCODE_RESET_MODE[7] |
| Grinding frames: $P_GFRAMES | Are retained, depending on:  
- MD20110 $MC_RESET_MODE_MASK  
- MD20152 $MC_GCODE_RESET_MODE[@[@]] |
### Frame conditions after RESET / part program end

<table>
<thead>
<tr>
<th>Frame</th>
<th>Condition after RESET / part program end</th>
</tr>
</thead>
</table>
| Complete basic frame: $P\_ACTB\_FRAME | Is retained, depending on:  
  - MD20110 $MC\_RESET\_MODE\_MASK bit 0 and bit 14  
  - MD10613 $MN\_NCBF\_RESET\_MASK  
  - MD24002 $MC\_CCHB\_FRAME\_RESET\_MASK  
| System frames:                 | Are retained, depending on:  
  - MD24006 $MC\_CHS\_FRAME\_RESET\_MASK  
  - MD20150 $MC\_GCODE\_RESET\_VALUES[ ]  
| $P\_PART\_FRAME, $P\_SET\_FRAME, $P\_ISO1\_FRAME, $P\_ISO2\_FRAME, $P\_ISO3\_FRAME, $P\_TOO\_FRAME, $P\_WP\_FRAME, $P\_TRA\_FRAME, $P\_ISO4\_FRAME, $P\_REL\_FRAME, $P\_CAC\_FRAME |  
| External work offset: $P\_EXT\_FRAME | Retained |
| DRF offset                     | Retained |

#### Deletion of system frames

The system frames in the data management can be deleted during RESET using machine data:

MD24007 $MC\_CHS\_FRAME\_RESET\_CLEAR\_MASK, bit<n> (deletion of system frames during RESET)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>System frame for actual value setting and scratching is deleted during RESET.</td>
</tr>
<tr>
<td>1</td>
<td>System frame for external work offset is deleted during RESET.</td>
</tr>
<tr>
<td>2</td>
<td>Reserved, for TCARR and PAROT see MD20150 $MC_GCODE_RESET_VALUES[ ].</td>
</tr>
<tr>
<td>3</td>
<td>Reserved, for TOROT and TOFRAME see MD20150 $MC_GCODE_RESET_VALUES[ ].</td>
</tr>
<tr>
<td>4</td>
<td>System frame for workpiece reference points is deleted during RESET.</td>
</tr>
<tr>
<td>5</td>
<td>System frame for cycles is deleted during RESET.</td>
</tr>
<tr>
<td>6</td>
<td>Reserved, RESET response depends on MD20110 $MC_RESET_MODE_MASK.</td>
</tr>
<tr>
<td>7</td>
<td>System frame for $P_ISO1_FR is deleted during RESET.</td>
</tr>
<tr>
<td>8</td>
<td>System frame for $P_ISO2_FR is deleted during RESET.</td>
</tr>
<tr>
<td>9</td>
<td>System frame for $P_ISO3_FR is deleted during RESET.</td>
</tr>
<tr>
<td>10</td>
<td>System frame for $P_ISO4_FR is deleted during RESET.</td>
</tr>
<tr>
<td>11</td>
<td>System frame $P_REL_FR is deleted during RESET.</td>
</tr>
</tbody>
</table>

### Part program start

#### Frame conditions after part program start

<table>
<thead>
<tr>
<th>Frame</th>
<th>Condition after part program start</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmable frame: $P_PFRAME</td>
<td>Deleted.</td>
</tr>
</tbody>
</table>
| Settable frames: $P\_IFRAME   | Retained depending on:  
  - MD20112 $MC\_START\_MODE\_MASK  

### Block search

#### Block search with calculation

Data management frames are also modified when carrying out a block search with calculation.

#### Cancellation of block search

If a block search is aborted with channel reset, then the following machine data can be used to set that all data management frames are reset to the value they had before the block search:

\[
\text{MD28560 } \text{S\text{MC\_MM\_SEARCH\_RUN\_RESTORE\_MODE}, bit<n>}
\]

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>All frames in the data management are restored.</td>
</tr>
</tbody>
</table>

In case of cascaded block searches, the frames are set to the status of the previous block search.

**SERUPRO**

The "SERUPRO" function is not supported.

### REPOS

There is no special treatment for frames. If a frame is modified in an ASUB, it is retained in the program. On repositioning with REPOS, a modified frame is included, provided the modification was activated in the ASUB.
10.6 Workpiece-related actual value system

10.6.1 Overview

Definition

The term "workpiece-related actual-value system" designates a series of functions that permit the user:

- To use a workpiece coordinate system defined in machine data after powerup.
  Features:
  - No additional operations are necessary.
  - Effective in JOG and AUTOMATIC modes
- To retain the valid settings for the following after end of program for the next part program:
  - Active plane
  - Settable frame (G54-G57)
  - Kinematic transformation
  - Active tool offset
- To change between work coordinate system and machine coordinate system via the user interface.
- To change the work coordinate system by operator action (e.g., changing the settable frame or the tool offset).

10.6.2 Use of the workpiece-related actual value system

Requirements, basic settings

The settings described in the previous Section have been made for the system.
The predefined setting after power-up of the HMI software is MCS.

Switchover to WCS

The change to the WCS via the user interface causes the axis positions relative to the origin of the WCS to be displayed.

Switchover to MCS

The change to the MCS via the user interface causes the axis positions relative to the origin of the MCS to be displayed.
Interrelationships between coordinate systems

The figure below shows the interrelationships between the machine coordinate system (MCS) and the workpiece coordinate system (WCS).

Figure 10-26 Interrelationship between coordinate systems

For further information, see "H2: Auxiliary function outputs to PLC (Page 377)" and "W1: Tool offset (Page 1363)".

References:

- Programming Guide Fundamentals
- Function Manual, Extended Functions; Kinematic Transformation (M1)
10.6.3 Special reactions

Overstore

Overstoring in `RESET` state of:

- Frames (zero offsets)
- Active plane
- Activated transformation
- Tool offset

immediately affects the actual-value display of all axes in the channel.

Entry via operator panel front

If operations on the operator panel are used to change the values for "Active frame" (zero offsets, "Parameters" operating area) and "Active tool length compensation" ("Parameters" operating area), one of the following actions is used to activate these changes in the display:

- Press the `RESET` key.
- Reselect:
  - Zero offset by the part program
  - Tool offset by the part program
- Reset:
  - Zero offset by overstoring
  - Tool offset by overstoring
- Part program start

MD9440

If the HMI machine data MD9440 ACTIVATE_SEL_USER_DATA for the operator panel front is set, the entered values become active immediately in `RESET` state.

When values are entered in the part-program execution stop state, they become effective when program execution continues.
Actual-value reading

If the actual value of $AA_IW$ is read in the WCS after activation of a frame (zero offset) or a tool offset, the activated changes are already contained in the result read even if the axes have not yet been traversed with the activated changes.

The actual values in the settable zero system (SZS) can be read from the part program for each axis using the variable $AA_IEN[axis]$.

The actual values in the basic zero system (BZS) can be read from the part program for each axis using the variable $AA_IBN[axis]$.

Actual-value display

The programmed contour is always displayed in the WCS.

The following offsets are added to the MCS:

- Kinematic transformation
- DRF offset/zero offset external
- Active frame
- Active tool offset of the current tool

Switchover by PLC

The actual values can be displayed in the WCS, SZS, BZS or MCS via the PLC. The PLC can define, which coordinate system corresponds to the workpiece coordinate system on a machine.

On MMC power-up the MCS is preset.
With the signal DB19 DBB0.7 "MCS/WCS switchover", it is also possible to switch from the PLC to the WCS.

Transfer to PLC

Depending on machine data MD20110 / MD20112, bit 1
, the auxiliary functions (D, T, M) are output to the PLC (or not) on selection of the tool length compensation.

Note

If the WCS is selected from the PLC, an operator action can still be used to switch between the WCS and MCS for the relevant mode.
However, when the mode and or area is changed, the WCS selected by the PLC is evaluated and activated (see Section "K1: Mode group, channel, program operation, reset response (Page 455)").
10.7 Restrictions

There are no supplementary conditions to note.

10.8 Examples

10.8.1 Axes

Axis configuration for a 3axis milling machine with rotary table

1. Machine axis: X1 Linear axis
2. Machine axis: Y1 Linear axis
3. Machine axis: Z1 Linear axis
4. Machine axis: B1 Rotary table (for turning for multiface machining)
5. Machine axis: W1 Rotary axis for tool magazine (tool revolver)
6. Machine axis: C1 (Spindle)

1. Geometry axis: X (1. channel)
2. Geometry axis: Y (1. channel)
3. Geometry axis: Z (1. channel)
1. Special axis: B (1. channel)
2. Special axis: WZM (1. channel)
1. Spindle: S1/C (1. channel)
### Parameterization of the machine data

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD10000 AXCONF_MACHAX_NAME_TAB[0]</td>
<td>= X1</td>
</tr>
<tr>
<td>MD10000 AXCONF_MACHAX_NAME_TAB[1]</td>
<td>= Y1</td>
</tr>
<tr>
<td>MD10000 AXCONF_MACHAX_NAME_TAB[2]</td>
<td>= Z1</td>
</tr>
<tr>
<td>MD10000 AXCONF_MACHAX_NAME_TAB[3]</td>
<td>= B1</td>
</tr>
<tr>
<td>MD10000 AXCONF_MACHAX_NAME_TAB[4]</td>
<td>= W1</td>
</tr>
<tr>
<td>MD10000 AXCONF_MACHAX_NAME_TAB[5]</td>
<td>= C1</td>
</tr>
<tr>
<td>MD20050 AXCONF_GEOAX_ASSIGN_TAB[0]</td>
<td>= 1</td>
</tr>
<tr>
<td>MD20050 AXCONF_GEOAX_ASSIGN_TAB[1]</td>
<td>= 2</td>
</tr>
<tr>
<td>MD20050 AXCONF_GEOAX_ASSIGN_TAB[2]</td>
<td>= 3</td>
</tr>
<tr>
<td>MD20060 AXCONF_GEOAX_NAME_TAB[0]</td>
<td>= X</td>
</tr>
<tr>
<td>MD20060 AXCONF_GEOAX_NAME_TAB[1]</td>
<td>= Y</td>
</tr>
<tr>
<td>MD20060 AXCONF_GEOAX_NAME_TAB[2]</td>
<td>= Z</td>
</tr>
<tr>
<td>MD20070 AXCONF_MACHAX_USED[0]</td>
<td>= 1</td>
</tr>
<tr>
<td>MD20070 AXCONF_MACHAX_USED[1]</td>
<td>= 2</td>
</tr>
</tbody>
</table>
### Machine data

<table>
<thead>
<tr>
<th>Machine Data</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD20070 AXCONF_MACHAX_USED[2]</td>
<td>= 3</td>
</tr>
<tr>
<td>MD20070 AXCONF_MACHAX_USED[3]</td>
<td>= 4</td>
</tr>
<tr>
<td>MD20070 AXCONF_MACHAX_USED[4]</td>
<td>= 5</td>
</tr>
<tr>
<td>MD20070 AXCONF_MACHAX_USED[5]</td>
<td>= 6</td>
</tr>
<tr>
<td>MD20080 AXCONF_CHANAX_NAME_TAB[0]</td>
<td>= X</td>
</tr>
<tr>
<td>MD20080 AXCONF_CHANAX_NAME_TAB[1]</td>
<td>= Y</td>
</tr>
<tr>
<td>MD20080 AXCONF_CHANAX_NAME_TAB[2]</td>
<td>= Z</td>
</tr>
<tr>
<td>MD20080 AXCONF_CHANAX_NAME_TAB[3]</td>
<td>= B</td>
</tr>
<tr>
<td>MD20080 AXCONF_CHANAX_NAME_TAB[4]</td>
<td>= WZM</td>
</tr>
<tr>
<td>MD20080 AXCONF_CHANAX_NAME_TAB[5]</td>
<td>= S1</td>
</tr>
<tr>
<td>MD30300 IS_ROT_AX[3]</td>
<td>= 1</td>
</tr>
<tr>
<td>MD30300 IS_ROT_AX[4]</td>
<td>= 1</td>
</tr>
<tr>
<td>MD30300 IS_ROT_AX[5]</td>
<td>= 1</td>
</tr>
<tr>
<td>MD30310 ROT_IS_MODULO[3]</td>
<td>= 1</td>
</tr>
<tr>
<td>MD30310 ROT_IS_MODULO[4]</td>
<td>= 1</td>
</tr>
<tr>
<td>MD30310 ROT_IS_MODULO[5]</td>
<td>= 1</td>
</tr>
<tr>
<td>MD30320 DISPLAY_IS_MODULO[3]</td>
<td>= 1</td>
</tr>
<tr>
<td>MD30320 DISPLAY_IS_MODULO[4]</td>
<td>= 1</td>
</tr>
<tr>
<td>MD20090 SPIND_DEF_MASTER_SPIND</td>
<td>= 1</td>
</tr>
<tr>
<td>MD35000 SPIND_ASSIGN_TO_MACHAX[AX1]</td>
<td>= 0</td>
</tr>
<tr>
<td>MD35000 SPIND_ASSIGN_TO_MACHAX[AX2]</td>
<td>= 0</td>
</tr>
<tr>
<td>MD35000 SPIND_ASSIGN_TO_MACHAX[AX3]</td>
<td>= 0</td>
</tr>
<tr>
<td>MD35000 SPIND_ASSIGN_TO_MACHAX[AX4]</td>
<td>= 0</td>
</tr>
<tr>
<td>MD35000 SPIND_ASSIGN_TO_MACHAX[AX5]</td>
<td>= 0</td>
</tr>
<tr>
<td>MD35000 SPIND_ASSIGN_TO_MACHAX[AX6]</td>
<td>= 1</td>
</tr>
</tbody>
</table>

### 10.8.2 Coordinate systems

#### Configuring a global basic frame

An NC with 2 channels is required. The following applies:

- The global basic frame can then be written by either channel.
- The other channel recognizes this change when the global basic frame is reactivated.
- The global basic frame can be read by either channel.
- Either channel can activate the global basic frame for that channel.

**Machine data**

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0]</td>
<td>= X1</td>
</tr>
<tr>
<td>MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1]</td>
<td>= X2</td>
</tr>
<tr>
<td>MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2]</td>
<td>= X3</td>
</tr>
<tr>
<td>MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3]</td>
<td>= X4</td>
</tr>
<tr>
<td>MD10000 $MN_AXCONF_MACHAX_NAME_TAB[4]</td>
<td>= X5</td>
</tr>
<tr>
<td>MD10000 $MN_AXCONF_MACHAX_NAME_TAB[5]</td>
<td>= X6</td>
</tr>
<tr>
<td>MD18602 $MN_MM_NUM_GLOBAL_BASE_FRAMES</td>
<td>= 1</td>
</tr>
<tr>
<td>MD28081 $MC_MM_NUM_BASE_FRAMES</td>
<td>= 1</td>
</tr>
</tbody>
</table>

**Machine data for channel 1**

<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MC_AXCONF_CHANAX_NAME_TAB[0] = X</td>
<td>$MC_AXCONF_CHANAX_NAME_TAB[0] = X</td>
</tr>
<tr>
<td>$MC_AXCONF_MACHAX_USED[0] = 1</td>
<td>$MC_AXCONF_MACHAX_USED[0] = 1</td>
</tr>
<tr>
<td>$MC_AXCONF_GEOAX_NAME_TAB[0] = X</td>
<td>$MC_AXCONF_GEOAX_NAME_TAB[0] = X</td>
</tr>
<tr>
<td>$MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1</td>
<td>$MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1</td>
</tr>
</tbody>
</table>

**Part program in first channel**

<table>
<thead>
<tr>
<th>Code (excerpt)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 $P_NCBFR[0] = CTRANS(x, 10) ; Activation of the NC global basic frame</td>
<td></td>
</tr>
<tr>
<td>. . .</td>
<td></td>
</tr>
<tr>
<td>N130 $P_NCBFRAME[0] = CROT(X, 45) ; Activation of the NC global basic frame with rotation =&gt; alarm 18310, since rotations of NC global frames are not permitted</td>
<td></td>
</tr>
<tr>
<td>. . .</td>
<td></td>
</tr>
</tbody>
</table>
### Part program in second channel

<table>
<thead>
<tr>
<th>Code (excerpt)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100 $P_{NCBFR}[0] = CTRANS( x, 10 ) ;</td>
<td>The NCU global basic frame is also active in second channel.</td>
</tr>
<tr>
<td>N510 G500 X10</td>
<td>Activate basic frame</td>
</tr>
<tr>
<td>N520 $P_{CHBFRAME}[0] = CTRANS( x, 10 ) ;</td>
<td>Current frame of second channel is activated with an offset.</td>
</tr>
</tbody>
</table>

### 10.8.3 Frames

**Example 1**

The channel axis is to become a geometry axis through geometry axis substitution. The substitution is to give the programmable frame a translation component of 10 in the X axis. The current settable frame is to be retained:

```
FRAME_GEOX_CHANGE_MODE = 1

$P_{UIFR}[1] = CROT(x,10,y,20,z,30) ; Frame is retained after geo axis substitution.
G54 ; Settable frame becomes active.
TRANS a10 ; Axial offset of a is also substituted.
GEOAX(1, a) ; a becomes x axis.
$P_ACTFRAME= CROT(x,10,y,20,z,30): CTRANS(x10)
```

Several channel axes can become geometry axes on a transformation change.

**Example 2**

Channel axes 4, 5 and 6 become the geometry axes of a 5axis orientation transformation. The geometry axes are thus all substituted before the transformation.

The current frames are changed when the transformation is activated.

The axial frame components of the channel axes, which become geometry axes, are taken into account when calculating the new WCS. Rotations programmed before the transformation are retained. The old WCS is restored when the transformation is deactivated.

The most common application will be that the geometry axes do not change before and after the transformation and that the frames are to stay as they were before the transformation.

**Machine data:**

```
$MN_FRAME_GEOAX_CHANGE_MODE = 1

$MC_AXCONF_CHANAX_NAME_TAB[0] = "CAX"
```
$MC\_AXCONF\_CHANAX\_NAME\_TAB[1] = \"CAY\"
$MC\_AXCONF\_CHANAX\_NAME\_TAB[2] = \"CAZ\"
$MC\_AXCONF\_CHANAX\_NAME\_TAB[3] = \"A\"
$MC\_AXCONF\_CHANAX\_NAME\_TAB[4] = \"B\"
$MC\_AXCONF\_CHANAX\_NAME\_TAB[5] = \"C\"

$MC\_AXCONF\_GEOAX\_ASSIGN\_TAB[0] = 1
$MC\_AXCONF\_GEOAX\_ASSIGN\_TAB[1] = 2
$MC\_AXCONF\_GEOAX\_ASSIGN\_TAB[2] = 3

$MC\_AXCONF\_GEOAX\_NAME\_TAB[0] = \"X\"
$MC\_AXCONF\_GEOAX\_NAME\_TAB[1] = \"Y\"
$MC\_AXCONF\_GEOAX\_NAME\_TAB[2] = \"Z\"

$MC\_TRAFO\_GEOAX\_ASSIGN\_TAB\_1[0] = 4
$MC\_TRAFO\_GEOAX\_ASSIGN\_TAB\_1[1] = 5
$MC\_TRAFO\_GEOAX\_ASSIGN\_TAB\_1[2] = 6

$MC\_TRAFO\_AXES\_IN\_1[0] = 4
$MC\_TRAFO\_AXES\_IN\_1[1] = 5
$MC\_TRAFO\_AXES\_IN\_1[2] = 6
$MC\_TRAFO\_AXES\_IN\_1[3] = 1
$MC\_TRAFO\_AXES\_IN\_1[4] = 2

Program:

$P\_NCBFRAME[0] = \text{ctrans}(x,1,y,2,z,3,a,4,b,5,c,6)
$P\_CHBFRAME[0] = \text{ctrans}(x,1,y,2,z,3,a,4,b,5,c,6)

$P\_IFRAME = \text{ctrans}(x,1,y,2,z,3,a,4,b,5,c,6):\text{crot}(z,45)
$P\_PFRAME = \text{ctrans}(x,1,y,2,z,3,a,4,b,5,c,6):\text{crot}(x,10,y,20,z,30)

\text{TRAOFI}
; \text{Geo axis (4,5,6) sets transformer}
; \quad \text{$P\_NCBFRAME[0] = \text{ctrans}(x,4,y,5,z,6,cax,1,cay,2,caz,3)$}
; \quad \text{$P\_ACTBFRAME = \text{ctrans}(x,8,y,10,z,12,cax,2,cay,4,caz,6)$}
; \quad \text{$P\_IFRAME = \text{ctrans}(x,4,y,5,z,6,cax,1,cay,2,caz,3):\text{crot}(z,45)$}
; \quad \text{$P\_PFRAME = \text{ctrans}(x,4,y,5,z,6,cax,1,cay,2,caz,3):\text{crot}(x,10,y,20,z,30)$}

\text{TRAOFI}
; \text{Geo axis (1,2,3) sets transformation deactivation}
; \quad \text{$P\_NCBFRAME[0] = \text{ctrans}(x,1,y,2,z,3,a,4,b,5,c,6)$}
; \quad \text{$P\_CHBFRAME[0] = \text{ctrans}(x,1,y,2,z,3,a,4,b,5,c,6)$}
10.9  Data lists

10.9.1  Machine data

10.9.1.1  Displaying machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MM__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SINUMERIK Operate</td>
<td></td>
</tr>
<tr>
<td>9242</td>
<td>MA_STAT_DISPLAY_BASE</td>
<td>Numerical basis for display of moving joint STAT</td>
</tr>
<tr>
<td>9243</td>
<td>MA_TU_DISPLAY_BASE</td>
<td>Numerical basis for display of rotary axis position TU</td>
</tr>
<tr>
<td>9244</td>
<td>MA_ORIAxes_EULER_ANGLE_NAME</td>
<td>Display of orientation axes as Euler angle</td>
</tr>
<tr>
<td>9245</td>
<td>MA_PRESET_FRAMEIDX</td>
<td>Value storage scratching and PRESET</td>
</tr>
<tr>
<td>9247</td>
<td>USER_CLASS_BASE_ZERO_OFF_PA</td>
<td>Availability of basic offset in &quot;Parameters&quot; operating area</td>
</tr>
<tr>
<td>9248</td>
<td>USER_CLASS_BASE_ZERO_OFF_MA</td>
<td>Availability of basic offset in Machine operating area</td>
</tr>
<tr>
<td>9424</td>
<td>MA_COORDINATE_SYSTEM</td>
<td>Coordinate system for actualvalue display</td>
</tr>
<tr>
<td>9440</td>
<td>ACTIVE_SEL_USER_DATA</td>
<td>Active data (frames) are immediately operative after editing</td>
</tr>
<tr>
<td>9449</td>
<td>WRITE_TOA_LIMIT_MASK</td>
<td>Applicability of MD9203 to edge data and locationdependent offsets</td>
</tr>
<tr>
<td>9450</td>
<td>MM_WRITE_TOA_FINE_LIMIT</td>
<td>Limit value for wear fine</td>
</tr>
<tr>
<td>9451</td>
<td>MM_WRITE_ZOA_FINE_LIMIT</td>
<td>Limit value for offset fine</td>
</tr>
</tbody>
</table>

10.9.1.2  NC-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>AXCONF_MACHAX_NAME_TAB</td>
<td>Machine axis name</td>
</tr>
<tr>
<td>10600</td>
<td>FRAME_ANGLE_INPUT_MODE</td>
<td>Sequence of rotation in the frame</td>
</tr>
<tr>
<td>10602</td>
<td>FRAME_GEOAX_CHANGE_MODE</td>
<td>Frames and switchover of geometry axes</td>
</tr>
<tr>
<td>10610</td>
<td>MIRROR_REF_AX</td>
<td>Reference axis for mirroring</td>
</tr>
<tr>
<td>10612</td>
<td>MIRROR_TOGGLE</td>
<td>Change over mirror</td>
</tr>
</tbody>
</table>
### K2: Axis Types, Coordinate Systems, Frames

#### 10.9 Data lists

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10613</td>
<td>NCBFRAME_RESET_MASK</td>
<td>Active NCU–global basic frame after reset</td>
</tr>
<tr>
<td>10615</td>
<td>NCBFRAME_POWERON_MASK</td>
<td>Reset global basic frames after Power On</td>
</tr>
<tr>
<td>10617</td>
<td>FRAME_SAVE_MASK</td>
<td>Behavior of frames for SAVE subprograms</td>
</tr>
<tr>
<td>10650</td>
<td>IPO_PARAM_NAME_TAB</td>
<td>Name of interpolation parameters</td>
</tr>
<tr>
<td>10660</td>
<td>INTERMEDIATE_POINT_NAME_TAB</td>
<td>Name of intermediate point coordinates for $G2/G3$</td>
</tr>
<tr>
<td>11640</td>
<td>ENABLE_CHAN_AX_GAP</td>
<td>Channel axis gaps are allowed</td>
</tr>
<tr>
<td>18600</td>
<td>MM_FRAME_FINE_TRANS</td>
<td>Fine offset for FRAME (SRAM)</td>
</tr>
<tr>
<td>18601</td>
<td>MM_NUM_GLOBAL_USER_FRAMES</td>
<td>Number of globally predefined user frames (SRAM)</td>
</tr>
<tr>
<td>18602</td>
<td>MM_NUM_GLOBAL_BASE_FRAMES</td>
<td>Number of global basic frames (SRAM)</td>
</tr>
</tbody>
</table>

#### 10.9.1.3 Channel-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20050</td>
<td>AXCONF_GEOAX_ASSIGN_TAB</td>
<td>Assignment geometry axis to channel axis</td>
</tr>
<tr>
<td>20060</td>
<td>AXCONF_GEOAX_NAME_TAB</td>
<td>Geometry axis name in channel</td>
</tr>
<tr>
<td>20070</td>
<td>AXCONF_MACHAX_USED</td>
<td>Machine axis number valid in channel</td>
</tr>
<tr>
<td>20080</td>
<td>AXCONF_CHANAX_NAME_TAB</td>
<td>Channel axis name in the channel</td>
</tr>
<tr>
<td>20110</td>
<td>RESET_MODE_MASK</td>
<td>Definition of basic control settings after RESET / TP end</td>
</tr>
<tr>
<td>20118</td>
<td>GEOAX_CHANGE_RESET</td>
<td>Allow automatic geometry axis change</td>
</tr>
<tr>
<td>20126</td>
<td>TOOL_CARRIER_RESET_VALUE</td>
<td>Active toolholder on RESET</td>
</tr>
<tr>
<td>20140</td>
<td>TRAFO_RESET_VALUE</td>
<td>Transformation record on power-up (RESET / TP-End)</td>
</tr>
<tr>
<td>20150</td>
<td>GCODE_RESET_VALUES</td>
<td>Initial setting of the G groups</td>
</tr>
<tr>
<td>20152</td>
<td>GCODE_RESET_MODE</td>
<td>RESET response of the G groups</td>
</tr>
<tr>
<td>20184</td>
<td>TOCARR_BASE_FRAME_NUMBER</td>
<td>Number of the basic frame for pickup of the table offset</td>
</tr>
<tr>
<td>21015</td>
<td>INVOLUTE_RADIUS_DELTA</td>
<td>End point monitoring for evolvents (involuteS)</td>
</tr>
<tr>
<td>22532</td>
<td>GEOAX_CHANGE_M_CODE</td>
<td>M code for replacement of geometry axes</td>
</tr>
<tr>
<td>22534</td>
<td>TRAFO_CHANGE_M_CODE</td>
<td>M code for transformation change</td>
</tr>
<tr>
<td>24000</td>
<td>FRAME_ADD_COMPONENTS</td>
<td>Frame components for G58 and G59</td>
</tr>
<tr>
<td>24002</td>
<td>CHBFrames_RESET_MASK</td>
<td>RESET response of channel-specific basic frames</td>
</tr>
<tr>
<td>24004</td>
<td>CHBFrames_POWERON_MASK</td>
<td>Reset channel-specific basic frames after Power On</td>
</tr>
<tr>
<td>24006</td>
<td>CHSFRAME_RESET_MASK</td>
<td>Active system frames after reset</td>
</tr>
<tr>
<td>24007</td>
<td>CHSFRAME_RESET_CLEAR_MASK</td>
<td>Clear system frames on RESET</td>
</tr>
<tr>
<td>24008</td>
<td>CHSFRAME_POWERON_MASK</td>
<td>Reset system frames after POWER ON</td>
</tr>
<tr>
<td>24010</td>
<td>PFrames_RESET_MODE</td>
<td>RESET mode for programmable frame</td>
</tr>
<tr>
<td>24020</td>
<td>FRAME_SUPPRESS_MODE</td>
<td>Positions for frame suppression</td>
</tr>
<tr>
<td>24030</td>
<td>FRAME_ACT_SET</td>
<td>SZS coordinate system setting</td>
</tr>
<tr>
<td>24040</td>
<td>FRAME_ADAPT_MODE</td>
<td>Adapting active frames</td>
</tr>
<tr>
<td>24050</td>
<td>FRAME_SAA_MODE</td>
<td>Saving and activating data management frames</td>
</tr>
<tr>
<td>24805</td>
<td>TRACYL_ROT_AX_FRAME_1</td>
<td>Rotary axis offset TRACYL 1</td>
</tr>
<tr>
<td>24855</td>
<td>TRACYL_ROT_AX_FRAME_2</td>
<td>Rotary axis offset TRACYL 2</td>
</tr>
</tbody>
</table>
### 10.9 Data lists

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC____</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>24905</td>
<td>TRANSMIT_ROT_AX_FRAME_1</td>
<td>Rotary axis offset TRANSMIT1</td>
</tr>
<tr>
<td>24955</td>
<td>TRANSMIT_ROT_AX_FRAME_2</td>
<td>Rotary axis offset TRANSMIT2</td>
</tr>
<tr>
<td>28080</td>
<td>MM_NUM_USER_FRAMES</td>
<td>Number of settable Frames (SRAM)</td>
</tr>
<tr>
<td>28081</td>
<td>MM_NUM_BASE_FRAMES</td>
<td>Number of basic frames</td>
</tr>
<tr>
<td>28082</td>
<td>MM_SYSTEM_FRAME_FRAMES</td>
<td>System frames (SRAM)</td>
</tr>
<tr>
<td>28560</td>
<td>MM_SEARCH_RUN_RESTORE_MODE</td>
<td>Restore data after a simulation</td>
</tr>
</tbody>
</table>

#### 10.9.1.4 Axis/spindlespecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA____</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>32074</td>
<td>FRAME_OR_CORRPOS_NOTALLOWED</td>
<td>FRAME or HL offset is not permitted</td>
</tr>
<tr>
<td>35000</td>
<td>SPIND_ASSIGN_TO_MACHAX</td>
<td>Assignment spindle to machine axis</td>
</tr>
</tbody>
</table>

### 10.9.2 Setting data

#### 10.9.2.1 Channelspecific setting data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $SC____</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>42440</td>
<td>FRAME_OFFSET_INCR_PROG</td>
<td>Work offsets in frames</td>
</tr>
<tr>
<td>42980</td>
<td>TOFRAME_MODE</td>
<td>Frame definition for TOFRAME, TOROT and PAROT</td>
</tr>
</tbody>
</table>

### 10.9.3 System variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA_ETRANS[axis]</td>
<td>External work offset</td>
</tr>
<tr>
<td>$AA_IBN[axis]</td>
<td>Actual value in basic zero coordinate system (BZS)</td>
</tr>
<tr>
<td>$AA_IEN[axis]</td>
<td>Actual value in settable zero point coordinate system (SZS)</td>
</tr>
<tr>
<td>$AA_OFF[axis]</td>
<td>Overlaid motion for programmed axis</td>
</tr>
<tr>
<td>$AC_DRF[axis]</td>
<td>Handwheel override of an axis</td>
</tr>
<tr>
<td>$AC_JOG_COORD</td>
<td>Coordinate system for manual traversing</td>
</tr>
<tr>
<td>$P_ACSFRAME</td>
<td>Active frame between BCS and SZS</td>
</tr>
<tr>
<td>$P_ACTBFRAME</td>
<td>Active complete basic frame</td>
</tr>
<tr>
<td>$P_ACTFRAME</td>
<td>Active complete frame</td>
</tr>
<tr>
<td>$P_BFRAME</td>
<td>1st active basic frame. Corresponds to $P_CHBFRAME[0]</td>
</tr>
<tr>
<td>$P_CHBFR[&lt;n&gt;]</td>
<td>Data management frame: Basic frame can be activated via G500, G54...G599</td>
</tr>
<tr>
<td>Identifier</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>$P_{CHBFRAME}[&lt;n&gt;]</td>
<td>Active basic frame</td>
</tr>
<tr>
<td>$P_{CHBFRMASK}</td>
<td>Basic frame mask in the channel</td>
</tr>
<tr>
<td>$P_{CHSFRMASK}</td>
<td>System frame mask</td>
</tr>
<tr>
<td>$P_{CYCFR}</td>
<td>Data management frame: System frame for cycles</td>
</tr>
<tr>
<td>$P_{CYCFRAME}</td>
<td>Active system frame for cycles</td>
</tr>
<tr>
<td>$P_{EXTFR}</td>
<td>Data management frame: System frame for work offset external</td>
</tr>
<tr>
<td>$P_{EXTFRAME}</td>
<td>Active system frame for external work offset</td>
</tr>
<tr>
<td>$P_IFRAME</td>
<td>Active settable frame</td>
</tr>
<tr>
<td>$P_ISO1FR</td>
<td>Data management frame: System frame for ISO G51.1 mirroring</td>
</tr>
<tr>
<td>$P_ISO2FR</td>
<td>Data management frame: System frame for ISO G68 2DROT</td>
</tr>
<tr>
<td>$P_ISO3FR</td>
<td>Data management frame: System frame for ISO G68 3DROT</td>
</tr>
<tr>
<td>$P_ISO4FR</td>
<td>Data management frame: System frame for ISO G51 Scale</td>
</tr>
<tr>
<td>$P_ISO1FRAME</td>
<td>Active system frame for ISO G51.1 Mirroring</td>
</tr>
<tr>
<td>$P_ISO2FRAME</td>
<td>Active system frame for ISO G68 2DROT</td>
</tr>
<tr>
<td>$P_ISO3FRAME</td>
<td>Active system frame for ISO G68 3DROT</td>
</tr>
<tr>
<td>$P_ISO4FRAME</td>
<td>Active system frame for ISO G51 Scale</td>
</tr>
<tr>
<td>$P_NCBFR[n]</td>
<td>Data management frame: NCU-global basic frame in the data management, can be activated via G500, G54...G599</td>
</tr>
<tr>
<td>$P_NCBFRAME[n]</td>
<td>Active NCU-global basic frame</td>
</tr>
<tr>
<td>$P_NCBFRMASK</td>
<td>Global basic frame mask</td>
</tr>
<tr>
<td>$P_PARTFR</td>
<td>Data management frame: System frame for TCARR and PAROT</td>
</tr>
<tr>
<td>$P_PARTFRAME</td>
<td>Active system frame for TCARR and PAROT with orientable toolholder</td>
</tr>
<tr>
<td>$P_PFRAME</td>
<td>Active programmable frame</td>
</tr>
<tr>
<td>$P_SETFR</td>
<td>Data management frame: System frame for actual value setting</td>
</tr>
<tr>
<td>$P_SETFRAME</td>
<td>Active system frame for actual value setting</td>
</tr>
<tr>
<td>$P_TOOLFR</td>
<td>Data management frame: System frame for TOROT and TOFRAME</td>
</tr>
<tr>
<td>$P_TOOLFRAME</td>
<td>Active system frame for TOROT and TOFRAME</td>
</tr>
<tr>
<td>$P_TRAFRAME</td>
<td>Data management frame: System frame for transformations</td>
</tr>
<tr>
<td>$P_TRAFRAME</td>
<td>Active system frame for transformations</td>
</tr>
<tr>
<td>$P_UBFR</td>
<td>1st basic frame in the channel in the data management that is activated after G500, G54...G599. Corresponds to $P_{CHBFR}[0].</td>
</tr>
<tr>
<td>$P_UIFR[n]</td>
<td>Data management frame: Settable frame, can be activated via G500, G54 to G599</td>
</tr>
<tr>
<td>$P_UIFRNUM</td>
<td>Number of active settable frame</td>
</tr>
<tr>
<td>$P_WPFR</td>
<td>Data management frame: System frame for the workpiece</td>
</tr>
<tr>
<td>$P_WPFRAME</td>
<td>Active system frame for the workpiece</td>
</tr>
</tbody>
</table>
10.9.4 Signals

10.9.4.1 Signals from channel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>T function modification</td>
<td>DB21, ... .DBX61.0-2</td>
<td>-</td>
</tr>
<tr>
<td>D function modification</td>
<td>DB21, ... .DBX62.0-2</td>
<td>-</td>
</tr>
<tr>
<td>T function 1</td>
<td>DB21, ... .DBB118-119</td>
<td>DB250x.DBD2000</td>
</tr>
<tr>
<td>D function 2</td>
<td>DB21, ... .DBB129</td>
<td>DB250x.DBD5000</td>
</tr>
<tr>
<td>Number of active function G group 1 (bit int)</td>
<td>DB21, ... .DBB208</td>
<td>DB350x.DBB0</td>
</tr>
<tr>
<td>Number of active function G group 2 (bit int)</td>
<td>DB21, ... .DBB209</td>
<td>DB350x.DBB1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Number of active function G group 29 (bit int)</td>
<td>DB21, ... .DBB236</td>
<td>DB350x.DBB28</td>
</tr>
</tbody>
</table>

10.9.4.2 Signals to axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept zero offset external</td>
<td>DB31, ... .DBX3.0</td>
<td>-</td>
</tr>
</tbody>
</table>

10.9.4.3 Signals from axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle/no axis</td>
<td>DB31, ... .DBX60.0</td>
<td>DB390x.DBX0.0</td>
</tr>
</tbody>
</table>
11.1 Brief Description

Function

The control system supports the machine manufacturer in implementing an emergency stop function on the basis of the following functions:

- An emergency stop button is installed in a location easily accessible to the machine operator on all SINUMERIK machine control panels. The functionality of the emergency stop button includes the positive opening of electrical switching contacts and a mechanical self-activating latching/locking.

- The emergency stop request to the NC is transmitted via the NC/PLC interface on the PLC.

- The Emergency Stop function must bring the machine to a standstill according to stop category 0 or 1 (EN 60204).

- In the case of an emergency stop, all machine functions controlled by the PLC can be brought to a safe state that can be set by the machine manufacturer.

- Unlatching the emergency stop button does not cancel the emergency stop state nor does it initiate a restart.

- After the emergency stop state has been canceled, it is not necessary to reference the machine axes or synchronize the spindles. The actual positions of the machine axes are continuously tracked during the emergency stop sequence.

11.2 Relevant standards

Relevant standards

Compliance with the following standards is essential for the emergency stop function:

- EN ISO 12000-1
- EN ISO 12000-2
- EN 418
- EN 60204

Emergency stop

In accordance with EN 418, an emergency stop is a function that:

- Is intended to prevent or diminish developing or existing risks to operating personnel, and damage to the machine or machined materials.

- Is triggered by a single action of a person, if the normal stop function is not suitable for it.
Hazards

In the terms of EN 418, risks may arise from:

- Functional irregularities (machine malfunctions, unacceptable properties of the material to be machined, human error, etc.).
- Normal operation.

Standard EN ISO 12000-2

In accordance with the basic safety requirement of the EC Machinery Directive regarding emergency stop, machines must be equipped with an emergency stop device.

Exceptions

No emergency stop device is required on machines:

- Where an emergency stop device would not reduce the risk, either because the shutdown time would not be reduced or because the measures to be taken would not be suitable for controlling the risk.
- That are held and operated manually.

Note

The machine manufacturer is expressly directed to comply with the national and international standards. The SINUMERIK controllers support the machine manufacturer in the implementation of the emergency stop function according to the specifications in the following function description. But the responsibility for the emergency stop function (its triggering, sequence and acknowledgement) rests exclusively with the machine manufacturer.

11.3 Emergency stop control elements

Emergency stop control elements

In accordance with EN 418, emergency stop control elements must be designed so that they latch mechanically on their own and are easy for the operator and others to actuate in the event of an emergency.

The following list includes some possible types of control elements:

- Mushroom pushbutton switches
- Wires/cables, cords, rods
- Puller grips
- In special cases: Foot switches without protective covers
Emergency stop button and control

Actuation of the emergency stop button or a signal derived directly from the button must be routed to the controller (PLC) as a PLC input. In the PLC user program, this PLC input must be forwarded to the NC on the interface signal:

DB10 DBX56.1 (Emergency stop)

Resetting of the emergency stop button or a signal derived directly from the button must be routed to the controller (PLC) as a PLC input. In the PLC user program, this PLC input must be forwarded to the NC on the interface signal:

DB10 DBX56.2 (Acknowledge emergency stop)

Connection conditions

For connecting the emergency stop button see:
References:
Operator Components Manual

11.4 Emergency stop sequence

After actuation of the emergency stop control element, the emergency stop device must operate in the best possible way to prevent or minimize the danger.

"In the best possible way" means that the most favorable delay rate can be selected and the correct stop category (defined in EN 60204) can be determined according to a risk assessment.

Emergency stop sequence in the NC

The predefined (in EN 418) sequence of internal functions implemented to obtain the emergency stop state is as follows in the control system:

1. Part program execution is interrupted.
   All machine axes are braked in the relevant axis-specific parameterized time:
   MD36610 $MA_AX_EMERGENCY_STOP_TIME (time of braking ramp in event of errors)
   The maximum braking ramp that can be achieved thereby, is defined by the maximum brake current of the respective drive. The maximum brake current is achieved by setting a setpoint = 0 (fast braking).

2. Reset interface signal:
   DB11 DBX6.3 (Mode group ready)

3. Set the interface signal:
   DB10 DBX106.1 (emergency stop active)

4. Alarm 3000 "Emergency stop" is displayed.
5. After the expiry of a parameterized delay time, the servo enables of machine axes are reset. The setting of the delay time is programmed in machine data: MD36620 $MA_SERVO_DISABLE_DELAY_TIME (OFF delay of the controller enable) The following setting rule must be observed: MD36620 ≥ MD36610

6. All machine axes are switched in the follow-up mode within the controller. The machine axes are no longer in position control.

Emergency stop sequence at the machine

The emergency stop sequence on the machine is determined solely by the machine manufacturer.

Attention should be paid to the following points in connection to the sequence on the NC:

- The process in the NC is started using the interface signal:
  DB10 DBX56.1 (Emergency stop)
  After the machine axes have come to a standstill, the power supply must be interrupted, in compliance with EN 418.

  Note
  The responsibility for interrupting the power supply rests with the machine manufacturer.

- The digital and analog outputs of the PLC I/O are not influenced by the emergency stop sequence in the NC.
  If individual outputs are required to attain a particular state or voltage level in the event of an emergency stop, the machine manufacturer must implement this in the PLC user program.

- The fast digital outputs of the NCK I/O system are not influenced by the emergency stop sequence in the NC.
  If individual outputs must assume a specific state in case of emergency stop, the machine manufacturer must transmit the desired state to the NC in the PLC user program via interface signals:
  DB10 DBB4-7

  Note
  If the sequence in the NC is not to be executed as described above, then the interface signal DB10 DBX56.1 (emergency stop) must not be set until an emergency stop state defined by the machine manufacturer in the PLC user program is reached.

  As long as the interface signal is not set and no other alarm is pending, all interface signals are operative in the NC. Any emergency stop state defined by the manufacturer (including axis-, spindle- and channel-specific emergency stop states) can therefore be assumed.

11.5 Emergency stop acknowledgement

The emergency stop control element may only be reset as a result of manual manipulation of the emergency stop control element according to EN 418.

Resetting of the emergency stop control element alone must not trigger a restart command.
A machine restart must be impossible until all of the actuated emergency stop control elements have been deliberately reset by hand.

**Emergency stop acknowledgement**

The EMERGENCY STOP state is only reset if the interface signal:DB10 DBX56.2 (acknowledge EMERGENCY STOP) is set followed by the interface signal:DB11, ... DBX0.7 (mode group reset).

Hence it can be noted that the interface signal DB10 DBX56.2 (acknowledge emergency stop) and the interface signal DB21, ... DBX7.7 (Reset) together are set at least for so long until the interface signal DB10 DBX106.1 (emergency stop active) is reset.

**Note**

The emergency stop state cannot be reset with the interface signal DB21, ... DBX7.7 (Reset) alone.

---

**Figure 11-1** Resetting the emergency stop state

1. DB10 DBX56.2 (acknowledge emergency stop) is inoperative
2. DB21, ... DBX7.7 (Reset) is inoperative
3. DB10 DBX56.2 and DB21, ... DBX7.7 reset DB10 DBX106.1 (emergency stop active)
Effects

Resetting the emergency stop state has the following effects:

- Within the controller for all machine axes:
  - The servo enables are set.
  - The follow-up mode is canceled.
  - The position control is activated.
- The following interface signals are set:
  DB31, DBX60.5 (position control active)
  DB11 DBX6.3 (mode group ready)
- The following interface signal is reset:
  DB10 DBX106.1 (emergency stop active)
- Alarm 3000 "Emergency stop" is deleted.
- Part program processing is interrupted in all channels of the NC.

PLC and NCK I/Os

The PLC user program must switch the PLC and NCK I/Os back to the state for operation of the machine.

POWER OFF / ON (supply off / on)

The emergency stop state can also be reset by switching the controller off and back on (POWER OFF / ON).

Requirement:

During power-up of the controller the interface signal DB10 DBX56.1 (emergency stop) must not be set.

11.6 Data lists

11.6.1 Machine data

11.6.1.1 Axis/spindlespecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>36610</td>
<td>AX_EMERGENCY_STOP_TIME</td>
<td>Length of the braking ramp for error states</td>
</tr>
<tr>
<td>36620</td>
<td>SERVO_DISABLE_DELAY_TIME</td>
<td>Cutout delay servo enable</td>
</tr>
</tbody>
</table>
### 11.6.2 Signals

#### 11.6.2.1 Signals to NC

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergency stop</td>
<td>DB10.DBX56.1</td>
<td>DB2600.DBX0.1</td>
</tr>
<tr>
<td>Acknowledge Emergency Stop</td>
<td>DB10.DBX56.2</td>
<td>DB2600.DBX0.2</td>
</tr>
</tbody>
</table>

#### 11.6.2.2 Signals from NC

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergency stop active</td>
<td>DB10.DBX106.1</td>
<td>DB2700.DBX0.1</td>
</tr>
</tbody>
</table>

#### 11.6.2.3 Signals to BAG

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode group RESET</td>
<td>DB11.DBX0.7</td>
<td>DB3000.DBX0.7</td>
</tr>
</tbody>
</table>
N2: Emergency stop

11.6 Data lists
12.1 Function

Transverse axis

Within the framework of “turning” technology, the transverse axis refers to the machine axis that travels perpendicular to the axis of symmetry of the spindle, in other words, to longitudinal axis Z.

Geometry axis as transverse axis

Every geometry axis of a channel can be defined as a transverse axis. For the geometry axis as a transverse axis, the following functions can be simultaneously or separately permitted and activated:

- Programming and display in the diameter
- Reference axis for constant cutting speed G96/G961/G962
Several transverse axes in the channel

The introduction of several transverse axes in the channel involves a functional decoupling of diameter programming and reference axis for G96/G961/G962. Diameter programming and reference axis for G96/G961/G962 can be active for different transverse axes (see table below).

<table>
<thead>
<tr>
<th>Permissible axis type:</th>
<th>Programming and display in the channel</th>
<th>Reference axis for G96 / G961 / G962</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry axis</td>
<td>Linear channel axes</td>
<td>Geometry</td>
</tr>
<tr>
<td>Selection in the channel:</td>
<td>one</td>
<td>m of 3</td>
</tr>
<tr>
<td>Specific effect:</td>
<td>Channel</td>
<td>Axis</td>
</tr>
<tr>
<td>Machine data:</td>
<td>MD20100</td>
<td>MD30460</td>
</tr>
<tr>
<td>Programming:</td>
<td>DIAM* channel-specific modal</td>
<td>axis-specific modal</td>
</tr>
<tr>
<td>Axis-specific non-modal diametral/radius programming:</td>
<td>DAC, DIC, RAC, RIC</td>
<td></td>
</tr>
</tbody>
</table>

DIAM*: DIAMOF, DIAMON, DIAM90, DIAMCYCOF
DIAM*A[<Axis>]: DIAMOFA[<Axis>], DIAMONA[<Axis>], DIAM90A[<Axis>], DIACYCOFA[<Axis>], DIAMCHANA[<Axis>]
<Axis>: Axis name for geometry, channel or machine axis name

Note

Rotary axes are not permitted to serve as transverse axes.

Diameter-related data

**DIAMON/DIAMONA[<Axis>]**

After activation of the diameter programming with DIAMON/DIAMONA[<Axis>] (see "Programming (Page 817)"), the following data refer to diameter dimensions:

- Display data of transverse axis in the workpiece coordinate system:
  - Setpoint and actual position
  - Distance-to-go
  - REPOS offset
- "JOG" mode:
  - Increments for incremental dimension (INC) and handwheel travel (dependent upon active MD)
• Part program programming:
  – End positions, independent of reference mode (G90 / G91)
  – Interpolation parameters of circular-path programming (G2 / G3) if these are programmed with part program instruction AC absolute.
• Actual values read with reference to the workpiece coordinate system (WCS):
  – $AA_{MW}[^{<\text{transverse axis}>}]$
    System variable of the measuring functions MEAS (measuring with delete distance-to-go) and MEAW (measuring without delete distance-to-go)
  – $P_{EP}[^{<\text{transverse axis}>}]$
  – $AA_{IW}[^{<\text{transverse axis}>}]$

**DIAM90/DIAM90A[^{<\text{Axis}>}]**

After activation of the reference-mode-dependent diameter programming with DIAM90/DIAM90A[^{<\text{Axis}>}], the following data are always displayed in relation to diameter regardless of the operating mode (G90 / G91):
  • Actual value
  • Actual values read with reference to the workpiece coordinate system (WCS):
    – $AA_{MW}[^{<\text{transverse axis}>}]$
      System variable of the measuring functions MEAS (measuring with delete distance-to-go) and MEAW (measuring without delete distance-to-go)
    – $P_{EP}[^{<\text{transverse axis}>}]$
    – $AA_{IW}[^{<\text{transverse axis}>}]$

**DIAMCYCOF/DIACYCOFA[^{<\text{Axis}>}]**

With DIAMCYCOF/DIACYCOFA[^{<\text{Axis}>}], a changeover to radius programming takes place within the controller. The diameter programming status that was active before DIAMCYCOF or DIACYCOFA[^{AX}] continues to be displayed to the HMI.

**Permanently radius-related data**

For transverse axes, the following data is always entered, programmed and displayed in relation to radius:
  • Part program programming
    – Interpolation parameters of circular-path programming with CIP
  • Offsets:
    – Tool offsets
    – Programmable and configurable frames
    – External work offset
    – DRF and preset offset
    – etc.
12.2 Parameterization

Defining a geometry axis as a transverse axis

Defining a geometry axis as a transverse axis in the channel is realized using machine data:

MD20100 $MC_DIAMETER_AX_DEF (geometry axis with transverse axis function)

Example:

MD20100 $MC_DIAMETER_AX_DEF="X" ; geometry axis X is the transverse axis in the channel.

For this axis, diameter programming and assigning a constant cutting speed with G96 / G961 / G962 are both permitted.

Several transverse axes in the channel

Defining additional transverse axes in the channel, for which the functionality of the axis-specific diameter programming is permitted, is performed using the machine data:

MD30460 BASE_FUNCTION_MASK (axis functions)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>Axis-specific diameter programming is not permitted.</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Axis-specific diameter programming is permitted.</td>
</tr>
</tbody>
</table>

Note

The setting MD30460 bit 2 = 1 is only possible for linear axes.
An axis can be simultaneously defined in MD20100 and in MD30460 (bit 2). For this, the channel-specific MD20100 has a higher priority than the axis-specific MD30460.

With MD20100, during power up, the transverse axis:

- is assigned the function G96/G961/G962.
- is assigned the channel-specific diameter programming DIAMON, DIAMOF, DIAM90, DIAMCYCOF.

After power up, this axis has the axis-specific basic position DIAMCHANA[<Axis>].

With MD30460 bit2, the additional enabling of the axis-specific operations DIAMONA[<Axis>], DIAMOFA[<Axis>], DIAM90A[<Axis>], DIACYCOFA[<Axis>], DIMCHANA[<Axis>] takes place.

### Dimensions for tool parameters

According to MD20360 $MC_TOOL_PARAMETER_DEF_MASK, for all of the transverse axes defined in the channel, the following tool parameters can be activated as diameter values:

<table>
<thead>
<tr>
<th>MD20360 $MC_TOOL_PARAMETER_DEF_MASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

Radius programming from MD20100 $MC_DIAMETER_AX_DEF and MD30460 $MA_BASE_FUNCTION_MASK bit 2 is taken into account as follows based on MD20360 $MC_TOOL_PARAMETER_DEF_MASK:

<table>
<thead>
<tr>
<th>MD20360 $MC_TOOL_PARAMETER_DEF_MASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### Behavior during handwheel travel

Behavior during handwheel travel (MD11346 $MN_HANDWH_TRUE_DISTANCE == 1 or 3) of a transverse axis while diameter programming is active can be adjusted with the following machine data:

**MD20624 $MC_HANDWH_CHAN_STOP_COND**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0</td>
<td>Only the half path of the specified increment is traveled.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>The specified increment is traveled completely.</td>
</tr>
</tbody>
</table>

### Displaying position values in the diameter

Position values of transverse axes are always displayed as diameter values, if:

**MD27100 $MC_ABSBLOCK_FUNCTION_MASK bit 0 = 1**

### Channel-specific basic position after power up, reset

The channel-specific basic position after power up / reset or the end of the parts program of G function group 29 (DIAMON, DIAM90, DIAMOF, DIAMCYCOF) is defined through machine data:

- **MD20150 $MC_GCODE_RESET_VALUE**
- Or (for MD20110 $MC_RESET_MODE_MASK bit 0 = 1):
  - **MD20152 $MC_GCODE_RESET_MODE**

The user can set the respective desired status via an event-controlled program call (prog-event).

If G96/G961/G962 is the basic position after power up, a transverse axis must be defined using MD20100 $MC_DIAMETER_AX_DEF, otherwise the alarm message 10870 is output.

Reference axis for G96/G961/G962 retained:

MD20110 $MC_RESET_MODE_MASK, bit 18=1 for reset or end of parts program

MD20112 $MC_START_MODE_MASK, bit 18=1 for start of parts program

---

**P1: Transverse axes**

**12.2 Parameterization**

---

**MD20360 $MC_TOOL_PARAMETER_DEF_MASK**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>Display of remaining path in WCS always as a radius</td>
</tr>
<tr>
<td>9</td>
<td>For all transverse axes, with MD11346 $MN_HANDWH_TRUE_DISTANCE==1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>• half of the path of the specified handwheel increment is traveled, if channel-specific or axis-specific diameter programming is active for this axis.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>• half of the path of the specified handwheel increment is always traveled.</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>When jogging around circles, the circle center point coordinate is always a radius value, see SD42690 $SC_JOG_CIRCLE_CENTRE</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>For cycle masks, the absolute values of the transverse axis are in the radius.</td>
</tr>
</tbody>
</table>
A reference axis for G96/G961/G962 can also be assigned without the application of a transverse axis in MD20100 via SCC[AX]. For this scenario, the constant cutting speed cannot be activated with G96. For more detailed information, see:

References:
Programming Manual Fundamentals, Feedrate Control and Spindle Motion
“Constant cutting rate (G96/G961/G962, G97/G971/G972, G973, LIMS, SCC)

12.3 Programming

Transverse axes can be programmed with respect to both diameter and radius. Generally, they are diameter-related, i.e. programmed with doubled path dimension so that the corresponding dimensional information can be transferred to the part program directly from the technical drawings.

Switching the diameter programming on/off

Channel-specific diameter programming
The activating or deactivating of the diameter programming is done via the modally active parts program statements of G function group 29:

- **DIAMON**: Diameter programming ON
- **DIAMOF**: Diameter programming OFF, in other words, radius programming ON
- **DIAM90**: Diameter or radius programming depending on the reference mode:
  - Diameter programming ON in connection with absolute dimensioning G90
  - Radius programming ON in connection with incremental dimensioning G91
- **DIAMCYCOF**: Radius programming for G90 and G91 ON; for the HMI, the last active G function of this group remains active

Reference is made exclusively to the transverse axis of the channel.
**12.4 Supplementary conditions**

**Axis-specific diameter programming for several transverse axes in one channel**

**Note**

The additionally specified axis must be activated via MD30460 $MA\_BASE\_FUNCTION\_MASK$ with bit 2 = 1.

The axis specified must be a known axis in the channel. Geometry, channel or machine axes are permitted.

Programming is not permitted in synchronized actions.

The following axis-specific modal statements can be programmed several times in a parts program block:

- **DIAMONA[<Axis>]**: Diameter programming for G90, G91, AC and IC ON
- **DIAMOFA[<Axis>]**: Diameter programming OFF, in other words, radius programming ON
- **DIAM90A[<Axis>]**: Diameter or radius programming depending on the reference mode:
  - Diameter programming ON in connection with absolute dimensioning G90 and AC
  - Radius programming ON in connection with incremental dimensioning G91 and IC
- **DIACYCOFA[Axis]**: Radius programming for G90 and G91 ON; for the HMI, the last active G function of this group remains active
- **DIAMCHANA[Axis]**: Acceptance of diameter programming channel status
- **DIAMCHAN**: All axes with MD30460, bit 2 = 1 accept the diameter programming channel status

Axis-specific modal statements have priority over the channel setting.

**12.4 Supplementary conditions**

**Axis replacement**

Due to a GET request from the parts program, the diameter programming status for an additional transverse axis is accepted in the new channel during axis replacement using RELEASE[<Axis>].

**Axis replacement in synchronized actions**

For axis replacement in synchronized actions, a transverse axis takes the status of the axis-specific diameter programming with it into the new channel if:

- with MD30460, bit 2 = 1 axis-specific diameter programming is permitted for the transverse axis.
- the transverse axis is not subordinated to the channel-specific diameter programming in the releasing channel.

The active dimension can be queried via the system variable $AA\_DIAM\_STAT[<Axis>]$. 
Axis replacement via axis container rotation

By rotating the axis container, the assignment of a channel axis can change to assignment of a machine axis. The current diameter programming status is retained however for the channel axis after the rotation. This also applies to the current channel and axis status, because the status is the same for all axes of the axis container at the time of the machine data "putting into effect" the status from MD30460 $MA_BASE_FUNCTION_MASK.

12.5 Examples

Example 1

X is a transverse axis defined via MD20100 $MC_DIAMETER_AX_DEF.

Y is a geometry axis and U is an additional axis. These two axes are additional transverse axes with diameter specifications defined in MD30460 $MA_BASE_FUNCTION_MASK with bit 2 = 1.

DIAMON is not active after power up.

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G0 G90 X100 Y50</td>
<td>;no diameter programming is active</td>
</tr>
<tr>
<td>N20 DIAMON</td>
<td>;Channel-specific diameter programming, in effect for X</td>
</tr>
<tr>
<td>N30 Y200 X200</td>
<td>;Dimensions: X in the diameter, Y in the radius</td>
</tr>
<tr>
<td>N40 DIAMONA[Y]</td>
<td>;axis-specific modal diameter programming, ;in effect for Y</td>
</tr>
<tr>
<td>N50 Y250 X300</td>
<td>;Dimensions: X and Y in diameter</td>
</tr>
<tr>
<td>N60 DIAM90</td>
<td>;Dimensions: X G90/AC in the diameter, G91/IC in the radius</td>
</tr>
<tr>
<td>N70 Y200</td>
<td>;Y: continuing, axis-specific modal diameter programming</td>
</tr>
<tr>
<td>N75 G91 Y20 U=DIC(40)</td>
<td>;Dimensions: Y in the diameter, U non-modally IC in the diameter</td>
</tr>
<tr>
<td>N80 X50 Y100</td>
<td>;Dimensions: X in the radius (G91), Y in the diameter</td>
</tr>
<tr>
<td>N85 G90 X100 U200</td>
<td>;Dimensions: X in the diameter, U in the radius</td>
</tr>
<tr>
<td>N90 DIAMCHANA[Y]</td>
<td>;Y accepts the channel status DIAM90</td>
</tr>
<tr>
<td>N95 G91 X100 Y100</td>
<td>;Dimensions: X and Y in the radius(G91)</td>
</tr>
<tr>
<td>N100 G90 X200 Y200</td>
<td>;Dimensions: X and Y in diameter</td>
</tr>
</tbody>
</table>

Example 2

Transverse axes with diameter specification applied as in the previous example.

X and Y are located in channel 1 and are also known in channel 2; i.e. permitted for axis replacement.

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel 1</td>
<td></td>
</tr>
<tr>
<td>N10 G0 G90 X100 Y50</td>
<td>;no diameter programming is active</td>
</tr>
<tr>
<td>N20 DIAMON</td>
<td>;Channel-specific diameter programming for X</td>
</tr>
</tbody>
</table>
Program code                      Comment
N30 Y200 X200                      ;Dimensions: X in the diameter, Y in the radius
N40 DIAMONA[Y]                     ;Y axis-specific modal diameter programming
N50 Y250 X300                      ;Dimensions: X and Y in diameter
N60 SETM(1)                        ;Synchronous marker 1
N70 WAIT(1,2)                      ;wait for synchronous marker 1 in channel 2
Channel 2                           ...
N50 DIAMOF                         ;channel 2 no diameter programming active
...                                 ...
N100 WAIT(1,1)                     ;wait for synchronous marker 1 in channel 1
N110 GETD(Y)                       ;Axis replacement direct Y
N120 Y100                           ;Y is the channel-specific diameter programming
                                         ;subordinated in channel 2; i.e. dimension in the radius

12.6 Data lists

12.6.1 Machine data

12.6.1.1 Channelspecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20050</td>
<td>AXCONF_GEOAX_ASSIGN_TAB[n]</td>
<td>Assignment of geometry axis to channel axis</td>
</tr>
<tr>
<td>20060</td>
<td>AXCONF_GEOAX_NAME_TAB[n]</td>
<td>Geometry axis name in channel</td>
</tr>
<tr>
<td>20100</td>
<td>DIAMETER_AX_DEF</td>
<td>Geometry axis with transverse axis function</td>
</tr>
<tr>
<td>20110</td>
<td>RESET_MODE_MASK</td>
<td>Definition of control basic setting after powerup and RESET / part program end</td>
</tr>
<tr>
<td>20112</td>
<td>START_MODE_MASK</td>
<td>Definition of the control basic settings for NC start</td>
</tr>
<tr>
<td>20150</td>
<td>GCODE_RESET_VALUES[n]</td>
<td>Reset G groups</td>
</tr>
<tr>
<td>20152</td>
<td>GCODE_RESET_MODE[n]</td>
<td>G code basic setting at RESET/end of parts program</td>
</tr>
<tr>
<td>20360</td>
<td>TOOL_PARAMETER_DEF_MASK</td>
<td>Definition of tool parameters</td>
</tr>
<tr>
<td>20624</td>
<td>HANDWH_CHAN_STOP_COND</td>
<td>Definition of the behavior of traveling with handwheel</td>
</tr>
<tr>
<td>27100</td>
<td>ABSBLOCK_FUNCTION_MASK</td>
<td>Parameterize block display with absolute values</td>
</tr>
</tbody>
</table>

12.6.1.2 Axis/spindlespecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>30460</td>
<td>BASE_FUNCTION_MASK</td>
<td>Axis functions</td>
</tr>
</tbody>
</table>
13.1 Brief description

General

The PLC basic program organizes the exchange of signals and data between the PLC user program and the NCK (Numerical Control Kernel), HMI (Human Machine Interface) and MCP (Machine Control Panel). In the case of signals and data, a distinction is made between the following groups:

- Cyclic signal exchange
- Event-driven signal exchange
- Messages

Cyclic signal exchange

The cyclically-exchanged signals consist primarily of bit arrays.

- They contain **commands** transferred from the PLC to the NCK (such as start or stop) and **status information** from the NCK (such as program running, interrupted, etc.).
- The bit fields are organized into signals for:
  - Mode group
  - Channels
  - Axes/spindles
  - General NCK signals

The cyclic exchange of data is performed by the basic program at the start of the PLC cycle (OB1). This ensures, for example, that the signals from the NCK remain constant throughout a cycle.

Event-driven signal exchange NCK → PLC

PLC functions that have to be executed as a function of the workpiece program are triggered by auxiliary functions in the workpiece program. If a block with auxiliary functions is executed, the type of auxiliary function determines whether the NCK has to wait for this function to execute (e.g. tool change) or whether the function will be executed together with the workpiece machining process (e.g. tool loading on milling machines with chain magazine).

Data transfer must be as fast and yet as reliable as possible, in order to minimize the effect on the NCK machining process. Data transfer is, therefore, interrupt- and acknowledgement-driven. The basic program evaluates the signals and data, acknowledges this to the NCK and transfers the data to the application interface at the start of the cycle. If the data does not require user acknowledgement, this does not affect NC processing.
**Event-driven signal exchange PLC → NCK**

An "event driven signal exchange PLC → NCK" takes place whenever the PLC transfers a request to the NCK (e.g. traversal of an auxiliary axis). In this case, data transfer is also acknowledgement-driven. When performed from the user program, this type of signal exchange is triggered using a function block (FB) or function call (FC).

The associated FBs (Function Blocks) and FCs (Function Calls) are supplied together with the basic program.

**Messages**

User messages are acquired and conditioned by the basic program. The message signals are transferred to the basic program via a specified bit array. where they are evaluated and, if message events occur, entered in the PLC's interrupt buffer by means of the ALARM S/SQ functions. If an HMI (e.g. SINUMERIK Operate) is being used, the messages are transferred to the HMI and displayed.

**PLC/HMI data exchange**

In this type of data exchange, the HMI takes the initiative, being referred to as the "client" on the bus system. The HMI polls or writes data. The PLC processes these requests at the cycle control point via the operating system. The PLC basic program is not involved in these exchanges.

**Note**

The function of the machine is largely determined by the PLC program. Every PLC program in the RAM can be edited with the programming device.

**Know-how protection for user blocks**

To protect the know-how contained in the the user blocks (OB, FB and FC), they can be encoded with the SBP tool (SIMATIC block protection) contained in SIMATIC STEP 7. These blocks can then no longer be opened, debugged and modified without specifying the password for the encoding.

When encoding, the automation system on whose PLC-CPU the blocks are to be executed, must be specified: SIMATIC and/or SINUMERIK PLC-CPU.

The handling of the blocks, e.g. loading to the CPU, is not affected by the encoding.

**Requirement**

SIMATIC STEP 7 as of Version 5.5 SP3
13.2 Key data of the PLC CPU

Key data of the PLC CPU

The overview of the key data of the PLC CPU integrated in the SINUMERIK NCU can be found in:

References
NCU 7x0.3 PN Manual, Section "Technical data"

Note
I/O addresses for integrated drives
The I/O addresses above 4096 are reserved for the integrated drives of the NCU and must not assigned otherwise.

Functions of the basic PLC program

<table>
<thead>
<tr>
<th>Scope</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Axes/spindles</td>
<td>31</td>
</tr>
<tr>
<td>Channels</td>
<td>10</td>
</tr>
<tr>
<td>Mode groups</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status/control signals</td>
</tr>
<tr>
<td>M decoders (M00-99)</td>
</tr>
<tr>
<td>G group decoders</td>
</tr>
<tr>
<td>Aux. function distributors</td>
</tr>
<tr>
<td>Aux. function transfer, interrupt-driven</td>
</tr>
<tr>
<td>M decoding acc. to list</td>
</tr>
<tr>
<td>Move axes/spindles from PLC</td>
</tr>
<tr>
<td>ASUB interface</td>
</tr>
<tr>
<td>Error/operating messages</td>
</tr>
<tr>
<td>Transfer MCP and HHU signals</td>
</tr>
<tr>
<td>Display control handheld unit</td>
</tr>
<tr>
<td>Read/write NCK variables and GUD</td>
</tr>
<tr>
<td>PI services</td>
</tr>
<tr>
<td>Tool management</td>
</tr>
<tr>
<td>Star/delta switchover</td>
</tr>
<tr>
<td>m:n</td>
</tr>
<tr>
<td>Safety Integrated</td>
</tr>
<tr>
<td>Program diagnostics</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3: Basic PLC program for SINUMERIK 840D sl</td>
</tr>
</tbody>
</table>
13.3 PLC operating system version

The PLC operating system version is displayed at:

- User interface of SINUMERIK Operate: "Operating area switchover" > "Diagnostics" > "Version" ⇒ version data / system software NCU: Selection "PLC" > "Details" ⇒ version data / system software NCU/PLC: The PLC operating system version is displayed in the first line is at "PLC 3xx...".

Note
The displayed version is SINUMERIK-specific. It is not compatible with the basic SIMATIC CPU.

- SIMATIC STEP 7, HW Config: In the properties of the PLC CPU in the SINUMERIK rack: "Properties - CPU 3xx..." > "Order no. / firmware": xxxx / Vx.y.z

Note
The version of the basic SIMATIC CPU is displayed.

13.4 PLC mode selector

The PLC mode selector is located on the front of the NCU module. The following PLC operating modes can be set via the PLC mode selector:

<table>
<thead>
<tr>
<th>S</th>
<th>Meaning</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RUN-P</td>
<td>The PLC program can be changed without activation of the password</td>
</tr>
<tr>
<td>1</td>
<td>RUN</td>
<td>Only read access operations are possible using a programming device (PG). It is not possible to make changes to the PLC program until the password has been set.</td>
</tr>
<tr>
<td>2</td>
<td>STOP</td>
<td>Processing the PLC program is stopped and all PLC outputs are set to substitute values.</td>
</tr>
<tr>
<td>3</td>
<td>MRES</td>
<td>The PLC is switched into the STOP state followed by a PLC general reset (default data).</td>
</tr>
</tbody>
</table>

1) Switch position of the PLC mode selector

References
A detailed description of the position of the PLC mode selector on the front of the NCU module, as well as its use in connection with NCK and PLC general reset can be found in:

CNC Commissioning Manual: NCK, PLC, Drive:

- Section "Switch-on/power-up" > "Operator control and display elements for power-up"
- Section "Switch-on/power-up" > "NCK and PLC general reset"
- Section "General tips" > "Separate NCK and PLC general reset"
13.5 Reserve resources (timers, counters, FC, FB, DB, I/O)

Reserve resources (timers, counters, FC, FB, DB, I/O)

The components below are reserved for the basic program:

- **Timer**
  No reservation

- **Counter**
  No reservation

- **FC, FB, DB**
  FC0 to FC29 and FB0 to FB29 are reserved for the basic program. The number range between 1000 and 1023 is also reserved for FCs and FBs. DB1 to DB62, DB71 to DB80 are reserved for data blocks. The number range 1000 to 1099 is also reserved in addition for DB. The data blocks of channels, axes/spindles and tool management functions that are not activated may be assigned as desired by the user.

- **I/O range**
  The PLC has an I/O address volume of 8192 bytes each for inputs and outputs. The address ranges starting at 4096 / 4096 are reserved for/occupied by integrated drives. However, diagnostic addresses for modules can be assigned to the highest address range as proposed by STEP 7. Furthermore, the address range between 256 and 287 is assigned for the NCK, CP and HMI in rack 0 on the SIMATIC 300 station.

13.6 Commissioning hardware configuration of the PLC CPU

The commissioning of the PLC CPU is described in detail in:

- **References**

  CNC Commissioning Manual: NCK, PLC, Drive:
  - Section: "Connect PG/PC to PLC"
  - Section: "Commissioning PLC"
  - Section: "Basics" > "PLC program"
  - Section: "General tips" > "Separate NCK and PLC general reset"
  - Section: "General tips" > "Integrating PG/PC into the network (NetPro)"

13.7 Starting up the PLC program

13.7.1 Installation of the basic program

The installation of the basic program is described in detail in:
References
CNC Commissioning Manual: NCK, PLC, Drive; Section: "Commissioning PLC" > "Creating a PLC program"

Note
Installation/update
Before installing the toolbox for SINUMERIK 840D sl, SIMATIC STEP 7 must be installed.
It is recommended that the hardware expansions for STEP 7 be installed again from the toolbox after an update of STEP 7.

Contents
The OB source programs, including standard parameterization, interface symbols and data-block templates for the handheld unit and M decoding functions are included in the basic program.

13.7.2 Application of the basic program
A new CPU program (e.g. "Turnma1") must be set up in a project by means of the STEP 7 software for each installation (machine).

Remark
The catalog structures of a project and the procedure for creating projects and user programs are described in the relevant SIMATIC documentation.

Procedure
The basic program blocks are copied using the SIMATIC Manager and "File" > "Open" > "Library".
The following components must be copied from the library:
- From the block container: FCs, FBs, DBs, OBs, SFC, SFB, UDT
- The source_files (from the source container): GPOB840D
- Possibly MDECLIST, HHU_DB and others
- The symbols table (from the symbols container)

Compatibility with STEP 7
There are no dependencies between the basic program and current STEP 7 versions.
13.7.3 Version codes

Basic program

The version of the basic program is displayed on the Version screen of the user interface along with the controller type.

The controller type is encoded as follows:

<table>
<thead>
<tr>
<th>Leftjustified decade of DB17.DB0 (byte 0)</th>
<th>Controller type</th>
</tr>
</thead>
<tbody>
<tr>
<td>03</td>
<td>SINUMERIK 840D sl (NCU 7x0)</td>
</tr>
</tbody>
</table>

User program

Users can also display their own PLC version codes in the Version screen. For this purpose, a data of type STRING containing a maximum of 54 characters must be defined in any data block. The data can contain a text of the user's choice. The parameterization on this string is done via a pointer on FB1. For this, the data block must be defined symbolically.

See also section "block descriptions" > "FB1: RUN_UP - basic program, start section (Page 914) ".

13.7.4 Machine program

The machine manufacturer creates the machine program using the library routines supplied with the basic program. The machine program contains the logic operations and sequences on the machine. The interface signals to the NC are also controlled in this program. More complex communication functions with the NCK, e.g. read/write NC data, tool-management acknowledgments, etc., are activated and executed via blocks FCs and FBs of the basic-program.

The machine program can be created in various STEP 7 creation languages, e.g. AWL, KOP, FUP, S7-HIGRAPH, S7GRAPH, SCL. The complete machine program must be generated and compiled in the correct sequence.

This means that blocks that are called by other blocks must generally be compiled before these blocks.

If blocks that are called by other blocks are subsequently modified in the interface (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) as the program is developed, then the call block and all blocks associated with it must be compiled again. This general procedure applies analogously to instance data blocks for FBs. If these sequence of operations is not maintained, time-stamp conflicts occur when the data retranslated into STEP 7. As such, the recompilability of the blocks is not ensured and with the function "Status of block" unnecessary conflicts can also appear. It is, moreover, advisable to generate blocks in ASCII/STL by means of the STEP 7 editor when they have been created in Ladder Diagram or in single statements (incremental mode).
13.7.5 Data backup

The PLC-CPU does not save any symbolic names, but instead only the datatype descriptions of the block parameters VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR and the datatypes of the global data blocks.

Note

No sensible recompilation is possible without the related project for this machine. This especially affects, for instance the function status of the block or the necessary changes done in the PLC-CPU programs later. It is, therefore, necessary to keep a backup copy of the STEP 7 project located in the PLC CPU on the machine. This is a great help for the service case and saves unnecessary consumption of time in restoring the original project.

If the STEP 7 project exists and has been created according to the instructions given above, then symbols can be processed in the PLCCPU on this machine. It may also be advisable to store the machine source programs as ".awl" files in case they are required for any future upgrade.

The source programs of all organization blocks and all instance data blocks should always be available.

13.7.6 PLC series startup, PLC archive

Once the blocks have been loaded to the PLC CPU, a series archive can be generated via the HMI user interface to back up data on the machine. To ensure data consistency, this backup must be created immediately after block loading when the PLC is in the Stop state. It does not replace the SIMATIC project backup as the series archive saves binary data only. For instance, no symbolic information is present here. In addition, no CPU DBs (SFC 22 DBs) or SDBs generated in the CPU are saved.

Selection of the SINUMERIK archiving program

The PLC series archive can be generated directly from the SIMATIC project as an alternative.

- Open the "Settings" dialog box in the SIMATIC Manager: Menu bar "Tools" > "Settings"
- Open the "Archive" tab
- Select the the SINUMERIK archiving program "SINUMERIK (*.arc)" in the "Preferred archiving program" drop-down list box.

Start of the SINUMERIK archiving program

The SINUMERIK archiving program is started in the SIMATIC Manager via the menu command "File" > "Archive".

The PLC archive is generated after assigning the archive name. If a project contains several program paths, the S7 program for which the PLC archive will be created can be selected in the dialog box. All blocks contained in the selected program path are archived (except data blocks created with SFC22 (online) in the CPU).

The "SDB archive" function can be activated or deactivated for the archiving program. If "SDB archive" is activated, a PLC archive is created that only contains the system data blocks (SDB) of the selected program path.
Automation

The process of generating a series archive can be automated (comparable to the command interface in STEP 7). In generating this series archive, the command interface is expanded. The following functions are available for this expansion:

The functions (shown here in VB script) are not available until server instantiations and Magic have been called:

Const S7BlockContainer = 1138689, S7PlanContainer = 17829889
Const S7SourceContainer = 1122308
set S7 = CreateObject("Simatic.Simatic.1")
Instantiate rem command interface of STEP 7
Set S7Ext = CreateObject("SimaticExt.S7ContainerExt")
Call S7Ext.Magic(""

Functions:

- Function Magic(bstrVal As String) As Long
- Function MakeSerienIB (FileName As String, Option As Long, Container As S7Container) As Long

Description

Function Magic(bstrVal As String) As Long

The call provides access to certain functions. The function must be called once after server instantiation. The value of bstrVal can be empty. This initiates a check of the correct STEP 7 version and path name in Autoexec. The functions are enabled with a return parameter of 0.

Return parameter (-1) = incorrect STEP 7 version
Return parameter (-2) = no entry in Autoexec.bat

Function MakeSeriesstart-up(FileName As String, Option As Long, Container As S7Container) As Long

"Option" parameter:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal series startup file with general reset</td>
</tr>
<tr>
<td>Bit 0 = 1:</td>
<td>Series startup file without general reset. When project contains SDBs, this option is inoperable. A general reset is then always executed</td>
</tr>
<tr>
<td>Bit 1 = 1:</td>
<td>Series startup file with PLC restart</td>
</tr>
</tbody>
</table>

Return parameter value:

<table>
<thead>
<tr>
<th>Return Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
</tr>
<tr>
<td>-1</td>
<td>Function unavailable, call Magic function beforehand</td>
</tr>
<tr>
<td>-2</td>
<td>File name cannot be generated</td>
</tr>
<tr>
<td>-4</td>
<td>Container parameter invalid or container block empty</td>
</tr>
<tr>
<td>-5</td>
<td>Internal error (memory request rejected by Windows)</td>
</tr>
</tbody>
</table>
Use in script

Program code

If S7Ext.Magic("") < 0 Then
  Wscript.Quit(1)
End If

Set Proj1 = s7.Projects("new")
set S7Prog = Nothing
Set s7prog = Proj1.Programs.Item(1) 'if there is only one program'

For Each cont In s7prog.Next
  If (Cont.ConcreteType = S7BlockContainer) Then
    ' Check block container
    Exit For
  End if
  Cont = Nothing
Next

Error = S7Ext.MakeSerienIB("f:\dh\arc.dir\PLC.arc", 0, Cont)
' Now error analysis

The For Each ... Next block programmed above can be programmed in the Delphi programming language as follows (the programming for C, C++ programming languages is similar):

Program code

Var
  EnumVar: IEnumVariant;
  rgvar: OleVariant;
  fetched: Cardinal;

  //For Each Next
  EnumVar := (S7Prog.Next._NewEnum) as IEnumVariant;
  While (EnumVar.Next(1,rgvar,fetched) = S_OK) Do Begin
    Cont := IS7Container(IDispatch(rgvar)); // block container
    Check sources
    If (Cont.ConcreteType = S7BlockContainer) Then Break;
    Cont := NIL;
  End;
13.7.7 Software upgrade

A general PLC reset should be performed to achieve a defined initial state before the PLC software is upgraded. In this case, among other things, all user data (program and data blocks) will be deleted. The PLC general reset is described in:

References:
Commissioning Manual CNC: NCK, PLC, Drive, General Tips,
Section: PLC general reset

Generating a new SIMATIC S7 project

In normal cases, the new PLC basic program is to be linked-in for a new NCU software version. The basic programs blocks must be loaded into the user project for this purpose. If the following program and data blocks are already in the user project, then these should not be transferred with the blocks of the basic PLC program: OB1, OB40, OB82, OB86, OB100, FC12 and DB4. These have been possibly changed by the user, and should not be overwritten. The new basic program must be linked with the user program. The following procedure must be taken into account:

1. Generate the text or source file of all user blocks before copying the basic PLC program.
2. Copy the new basic program blocks into the SIMATIC S7 project (for a description, see Section "Application of the basic program (Page 826)"
3. All user programs "*.awl" must be recompiled in the correct order! (see also: " Machine program (Page 827)")
4. This newly compiled SIMATIC S7 project should then be downloaded with STEP 7 into the PLC.

However, it is normally sufficient to recompile the organization blocks (OBs) and the instance data blocks of the S7 project. This means before upgrading, only the sources for the organization blocks and the instance data blocks have to be generated.

NC variables

The latest NC VAR selector can be used for each NC software version (even earlier versions). The variables can also be selected from the latest list for earlier NC software versions. The data content in DB120 (default DB for variables) does not depend on the software status. That is, variables selected in an older software version need not be reselected when the software is upgraded.

13.7.8 I/O modules (FM, CP modules)

Additional packages for STEP 7 are generally required for more complex I/O modules (FM, CP modules). Support blocks (FC/FB) are provided in these additional packets. The blocks contain specific functions for operating the relevant module. These functions can be parameterized and called in the user program.
Identical numbers

If handling and basic program blocks have identical numbers, the block numbers of the basic program must remain unchanged. The block numbers of the handling blocks must be renamed to free numbers via STEP 7.

13.7.9 Troubleshooting

This section describes problems which may occur, their causes and remedies and should be read carefully before hardware is replaced.

<table>
<thead>
<tr>
<th>Serial no.</th>
<th>Errors</th>
<th>Cause/description</th>
<th>To correct or avoid errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No connection via MPI to PLC.</td>
<td>The MPI cable is not plugged in or is defective. Possibly, the STEP 7 software is also not correctly configured for the MPI card.</td>
<td>Test: Create a link with the programmer in the STEP 7 editor by means of connection &quot;Direct_PLC&quot;. A number of node addresses must be displayed here. If they do not appear, the MPI cable is defective/not plugged in.</td>
</tr>
<tr>
<td>2</td>
<td>PLC cannot be accessed in spite of PLC general reset.</td>
<td>A system data block SDB 0 has been loaded with a modified MPI address. This has caused an MPI bus conflict due to dual assignment of addresses.</td>
<td>Disconnect all MPI cables to other components. Create the link &quot;Direct_PLC&quot; with the programmer. Correct the MPI address.</td>
</tr>
<tr>
<td>3</td>
<td>All four LEDs on the PLC flash (DI disaster)</td>
<td>A system error has occurred in the PLC. <strong>Measures:</strong> The diagnostic buffer on the PLC must be read to analyze the system error in detail. To access the buffer, the PLC must be stopped (e.g. set &quot;PLC&quot; switch to position 2). A hardware reset must then be performed. The diagnostic buffer can then be read out with STEP 7. Relay the information from the diagnostic buffer to the Hotline / Development Service. A general reset must be carried out if requested after the hardware RESET. The diagnostic buffer can then be read with the PLC in the Stop state.</td>
<td>Once the PLC program has been RESET or reloaded, the system may return to normal operation. Even in this case, the content of the diagnostic buffer should be sent to the Development Office.</td>
</tr>
</tbody>
</table>
13.8 Coupling of the PLC CPU

13.8.1 General

A CPU of the S7-300 automation system is used as PLC for the SINUMERIK 840D sl. The PLC-CPU is integrated into the NCU component as a sub-module. A reference to the performance data of the PLC CPU can be found in Section "Key data of the PLC CPU (Page 823)".

13.8.2 Properties of the PLC CPU

The PLC integrated in the SINUMERIK 840D sl generally has the same functionality as the corresponding SIMATIC S7-300 PLC.

For differences, see reference in Section "Key data of the PLC CPU (Page 823)".

Owing to differences in their memory system as compared to a SIMATIC S7-300 PLC, certain functions are not available (e.g. save blocks on memory card, save project on memory card).

**Note**

As with the PLC integrated in SINUMERIK, there is no automatic start of the PLC after power failure and recovery for a SIMATIC S7-300 PLC when a "PLC stop" is triggered by an operator action on the programming device. For safety reasons, the PLC remains in the stop state with an appropriate diagnostic entry. You can start the PLC only by an operator action on the programming device, "Execute a restart", or via the mode selector "Stop" > "Run" (warm restart).

13.8.3 Interface with integrated PLC

**Physical interfaces**

With the SINUMERIK 840D sl, the PLC integrated in the NCU offers the option of exchanging signals between the NCK and PLC directly via a dual-port RAM.

**Data exchange with the operator panel**

Data exchange with the operator panel (e.g. TCU/OP) can be performed via Ethernet or PROFIBUS. With a connection via Ethernet, communication takes place via the integrated communication processor (CP 840D sl).

Data exchange with the machine control panel (MCP) and handheld unit (HHU) can be performed via MPI, PROFIBUS or Ethernet.

Programming devices should preferably be connected via Ethernet or via MPI (Multi-Point Interface) directly to the PLC.
Interface: NCK/PLC

The data exchange between NCK and PLC is organized by the basic program on the PLC side. The status information, such as "Program running", stored by the NCK in the NCK/PLC interface is copied to data blocks by the basic program at the beginning of the cycle (OB1) and can then be accessed in the user program (user interface). The control signals for the NCK (e.g. NC start) entered in the interface data block by the user are also written to the internal DPR and transferred to the NCK at the start of the cycle.

Workpiece-program-specific auxiliary functions transferred to the PLC are first evaluated by the basic program (interrupt-driven) and then transferred to the user interface at the start of OB 1. If the relevant NC block contains auxiliary functions that require that NCK processing is interrupted (e.g. M06 for tool change), the basic program stops the decoding of the NCK block initially for one PLC cycle. The user can then use the "read disable" interface signal to halt the block execution until the tool change has been completed. If, on the other hand, the relevant NC block only contains auxiliary functions, which do not require interruption of the decoding.
(e.g. M08 for cooling medium on), the transfer of these "fast" auxiliary functions is directly acknowledged in OB40, so that decoding is only insignificantly influenced by the transfer to the PLC.

The evaluation and enabling of the G functions transferred from the NCK are also alarm-driven, however they are transferred directly to the user interface. Where a G function is evaluated at several points in the PLC program, differences in the information of the G function within one PLC cycle may arise.

In the case of NC actions triggered and assigned with parameters by the PLC (e.g. traverse concurrent axes), triggering and parameter assignment is performed using FCs and FBs, not interface data blocks. The FCs and FBs belonging to the actions are supplied together with the basic program. The FCs and FBs required must be loaded by the user and called in the PLC program of the machine manufacturer (machine program). For an overview of FC, FB and data blocks, sorted according to basic and extended functions, please refer to Section "Start-up of PLC programs".

Interface: HMI/PLC

HMI/PLC data exchange is performed via the integrated CP, whereby the HMI is always the active partner (client) and the PLC is always the passive partner (server). Data transferred or requested by the HMI is read from and written to the HMI/PLC interface area by the PLC operating system (timing: Cycle control point). From the viewpoint of the PLC application, the data is identical to I/O signals.

Interface: MCP/PLC or HHU/PLC (connection: Ethernet)

MCP/PLC and HHU (HT 2) / PLC data is exchanged via the integrated CP. The CP transfers the MCP/HHU signals to and fetches them from the PLC's internal DPR (Dual-Port RAM). On the PLC side, the basic program handles communication with the user interface. The basic-program parameters (FB1, DB7) define the operand areas (e.g. I/O areas) and the start addresses.

Interface: MCP/PLC (connection: PROFIBUS)

MCP/PLC data exchange takes place via the PLC's PROFIBUS. The MCP's I/O addresses are to be set in the PLC's process image area and via HW configuration in STEP 7. The MCP*In, MCP*Out pointer variables must be set to the same addresses. The selected DP slave number must be entered in MCP*BusAdr.

Interface: HHU/PLC (connection: MPI)

The HHU/PLC data exchange is performed via the MPI interface on the PLC. The "Communication with global data (GD)" service is used for this purpose (see also STEP 7 User Manual). The PLC operating system handles the transfer of signals from and to the user interface. The STEP 7 "Communication configuration" configuring tool is used to define both GD parameters as well as operand areas (e.g. I/O areas) and their start addresses.

13.8.4 Diagnostic buffer on PLC

The diagnostic buffer of the PLC (readable using STEP 7) will enter diagnostic information on the PLC operating system.
13.9 Interface structure

**Interface DBs**

Mapping in interface data blocks is necessary due to the large number of signals exchanged between the NCK and PLC. These are global data blocks from the viewpoint of the PLC program. During system start-up, the basic program creates these data blocks from current NCK machine data (no. of channels, axes, etc.). The advantage of this approach is that the minimum amount of PLC RAM required for the current machine configuration is used.

13.9.1 PLC/NCK interface

**General**

The PLC/NCK interface comprises a data interface on one side and a function interface on the other. The data interface contains status and control signals, auxiliary functions and G functions, while the function interface is used to transfer jobs from the PLC to the NCK.

**Data interface**

The data interface is subdivided into the following groups:

- NCK-specific signals
- Mode-group-specific signals
- Channel-specific signals
- Axis/spindle/drive-specific signals
Function interface

The function interface is formed by FBs and FCs. The figure below illustrates the general structure of the interface between the PLC and the NCK.

Figure 13-2  PLC/NCK user interface
Compile-cycle signals

In addition to the standard signals exchanged between the PLC and NCK, an interface data block for compile cycles is also generated if required (DB9). The associated signals, which are dependent on the compile cycles, are transmitted cyclically at the start of OB1. The basic program starts transmission at the lowest address and works up to the highest. First, signals are transferred from the PLC to the NCK, then from the NCK to the PLC. The user must synchronize the NCK and PLC as necessary (e.g. using the semaphore technique). Signal transmission is asynchronous between NCK and PLC. This means, for example, that active NCK data transmission can be interrupted by the PLC. This can mean that data is not always consistent.

PLC/NCK signals

The group of signals from the PLC to NCK includes:

- Signals for modifying the digital and analog I/O signals of the NCK
- Keyswitch and emergency stop signals

![PLC/NCK interface diagram]

Figure 13-3 PLC/NCK interface
NCK/PLC signals

The group of signals from the NCK to PLC includes:
- Actual values of the digital and analog I/O signals of the NCK
- Ready and status signals of the NCK

Also output in this group are the HMI handwheel selection signals and the status signals.

The signals for handwheel selection are decoded by the basic program and entered in the machine/axis-specific interface.

Digital/analog I/Os of the NCK

The following must be noted with respect to the digital and analog I/Os of the NCK:

Inputs:
- All input signals or input values of the NCK are also transferred to the PLC.
- The transfer of signals to the NC part program can be suppressed by the PLC. Instead, a signal or value can be specified by the PLC.
- The PLC can also transfer a signal or value to the NCK even if there is no hardware for this channel on the NCK side.

Outputs:
- All signals or values to be output are also transferred to the PLC.
- The NCK can also transfer signals or values to the PLC even if there is no hardware for this channel on the NCK side.
- The values transferred by the NCK can be overwritten by the PLC.
- Signals and values from the PLC can also be output directly via the NCK I/O devices.

Note

When realizing the digital and analog NCK I/Os the information contained in the following documentation must be taken into account:

References:
Functions Manual Extended Functions; Digital and analog NCK I/Os (A4)

Signals PLC/Mode group

The operating mode signals set by the machine control panel or the HMI are transferred to the operating mode group (BAG) of the NCK. These apply to all NCK channels. Several mode groups can be optionally defined in the NCK.

The mode group reports its current status to the PLC.
Signals PLC/NCK channels

The signal groups below must be considered on the interface:

- Control/status signals
- Auxiliary/G functions
- Tool management signals
- NCK functions

The control/status signals are transmitted cyclically at the start of OB1. The signals entered in the channel-specific interface by the HMI (HMI signals are entered by the PLC operating system) are also transferred at this time if they have been defined on the HMI operator panel, not on the MCP.

Auxiliary functions and G functions are entered in the interface data blocks in two ways. First, they are entered with the change signals.

- The M signals M00 - M99 (they are transferred from the NCK with extended address 0) are also decoded and the associated interface bits set for the duration of one cycle.
- For G functions, only the groups selected via machine data are entered in the interface data block.
- The S values are also entered together with the related M signals (M03, M04, M05) in the spindle-specific interface. The axis-specific feedrates are also entered in the appropriate axis-specific interface.

When the tool management (magazine management) function is activated in the NCK, the assignment of spindle or revolver and the loading/unloading points are entered in separate interface DBs (DB71-73)

The triggering and parameter assignment of NCK functions is performed by means of PLC function calls.

The following function calls are available:

- Position a linear axis or rotary axis
- Position an indexing axis
- Start a prepared asynchronous subprogram (ASUB)
- Reading/writing of NC variables
- Update magazine and tool motion
Some of the above functions are described in their own function documentation.

---

**PLC/axis, spindle, drive signals**

The axis-specific and spindle-specific signals are divided into the following groups:

- Shared axis/spindle signals
- Axis signals
- Spindle signals
- Drive signals

The signals are transmitted cyclically at the start of OB1 with the following exceptions:

Exceptions include:

- Axial F value
- M value
- S value
An axial F value is entered via the M, S, F distributor of the basic program if it is transferred to the PLC during the NC machining process.

The M and S value are also entered via the M, S, F distributor of the basic program if one or both values require processing.

![Diagram of PLC interface with axes/spindles/drives](image)

Figure 13-6  Interface between PLC and axes/spindles/drives

### 13.9.2 Interface PLC/HMI

#### General

The following groups of functions are required for the PLC/HMI interface:

- Control signals
- Machine operation
- PLC messages
- PLC status display
Control signals

Some control signals are signal inputs, for example, via the machine control panel, which have to be taken into account by the HMI. This group of signals includes, for example, display actual values in MCS or WCS, key disable, etc. These are exchanged with the HMI via a separate interface data block (DB19).

Machine operation

All operator inputs, which lead to response actions on the machine, are monitored by the PLC. Operator actions are usually performed on the machine control panel (MCP). However, it is also possible to perform some operator actions on the HMI, e.g. mode selection.

The PLC operating system enters the operating signals sent by the HMI directly into the interface data blocks. As standard, the basic program routes these operating signals in such a way that, provided equivalent operator actions are available, these can be performed either on the HMI or on the MCP. If required, the user can switch off operation via HMI using an FB1 parameter "MMCToIF".

PLC messages

The signaling functions are based on the system diagnostic functions integrated in the operating system of the AS 300. These have the following characteristics:

- The PLC operating system enters all important system states and state transitions in a diagnostics status list. Communication events and I/O module diagnostics data (for modules with diagnostic functions) are also entered.

- Diagnostics events, which lead to a system stop, are also entered with a time stamp in a diagnostic buffer (circular buffer) in the chronological order of their occurrence.

- The events entered in the diagnostic buffer are automatically transmitted to human machine interface systems (OP or HMI) via the bus systems once these have issued a ready signal (message service). "Transfer to the node ready" is a function of the PLC operating system. Receipt and interpretation of the messages are executed by the HMI software.

- The PLC user program can also use SFCs (System Function Calls) to enter messages in the diagnostic buffer or ALARM S/ALARM SQ buffer.

- The events are entered in the alarm buffer. The associated message texts must be stored on the OP or HMI.

An FC (FC10) for message acquisition is prepared in conjunction with the basic program. This FC records events, subdivides them into signal groups and reports them to the HMI via the alarm buffer.
The message acquisition structure is shown in the figure "Acquisition and signaling of PLC events". The features include:

- Bit fields for events related to the NC/PLC interface are combined in a single data block (DB2) with bit fields for user messages.
- Bit fields are evaluated at several levels by FC10.
  - **Evaluation 1; acquisition of group signals**
    A group signal is generated for each group of signals if at least one bit signal is set to "1". This signal is generally linked to the disable signal of the NC/PLC interface (on modules with diagnostic functions). The group signals are acquired completely in cycles.
  - **Evaluation 2; acquisition of alarm messages**
    A fixed specification exists to define which signals in a group generate an alarm message when they change from "0" to "1".
  - **Evaluation 3; acquisition of operating signals**
    A fixed specification exists to define which signals in a group generate an operational message.

- The scope of the user bit fields (user area) is set by default to 10 areas with 8 bytes each, but the number of areas can also be adjusted to suit the requirements of the machine manufacturer via basic program parameters in FB1.

**Acknowledgement concept**

The following acknowledgement procedures are implemented for error and operational messages:

- **Operating messages** are intended for the display of normal operating states as information for the user. Acknowledgement signals are, therefore, not required for this type of message. An entry is made in the diagnostic status list for incoming and outgoing messages. The HMI maintains an up-to-date log of existing operating messages using the identifiers "operating message arrived" and "operating message gone".

- **Alarm messages** display error states on the machine, which will usually lead to the machine being stopped. Where several errors occur in rapid succession, it is important to be able to distinguish their order of occurrence for troubleshooting purposes. This is indicated, on the one hand, by the order in which they are entered in the diagnostic buffer and on the other, by the time stamp, which is assigned to every entry.

If the cause of the error disappears, the associated alarm message is only deleted if the user has acknowledged it (e.g. by pressing a key on the MCP). In response to this signal, the "Message acquisition" FC examines which of the reported errors have disappeared and enters these in the diagnostic buffer with the entry "Alarm gone". This enables the HMI to also maintain an up-to-date log of pending alarm messages. The time of day indicating the time at which the error occurred is maintained for messages, which are still pending (in contrast to a received interrogation).

**STEP 7**

A tool can be started in the SIMATIC Manager via menu item "Target system" > "CPU messages". Alarms and messages can be displayed by number using this tool. To do this, activate the "Alarm" tab and enter a check mark under "A" in the upper half of the screen.
User program

The user PLC program merely needs to call the basic program block FC10 with appropriate parameter settings in the cyclic program section and set or reset the bit fields in DB2. All further necessary measures are implemented by the basic program and HMI.

![Diagram of PLC events]

Figure 13-7 Acquisition and signaling of PLC events

Extensions of the PLC alarms via block FC10

The FB1 parameter "ExtendAlMsg" selects the PLC alarm mechanism. If "ExtendAlMsg:= FALSE" the earlier process of the FC10 with the DB2 is active as bit array data block. The known restrictions regarding the number of channels and axes are applicable.

On the other hand, in case of "ExtendAlMsg:= TRUE" the extension of the FC10 becomes active. DB2 and DB3 are created just as before. The user must set or reset the bits in DB2. The parameter setting via message and alarm and a parameter setting of the numeric value of the 2nd decade of the user alarms are contained in DB5.
The extensions are:

- Support for 10 channels, 31 axes, 64 user areas (the number of user areas should be entered in the FB1 parameter "MsgUser").

- Areas for feed stop, read-in disable, etc. are available without messages. The information from this area is stored on the interface in DB21, DB31 depending on FC10 parameter "ToUserIF" together with the related message bits as group signals. As such, the previous cumbersome handling of the signals is omitted.

- The alarms / messages also get the 16-bit integer additional value (%Z parameter in the alarm text) in addition to the alarm number for the user area 0. The user must write the 16-bit integer values in the DB2 in the array variable ZInfo0 parallel to setting an alarm bit. An integer value is available for each bit in the user area 0, see UDT1002 in the basic program.

- The user messages can be parameteized in the second decade of the message number in the numerical range 0 to 9. The display value of the second decade must be written by the user in the DB5 in the array variable UserDek2No. A number can be defined for each user area, see DB5 in the basic program.

  The value 0 is set by default for the second decade.

The structuring of the DB2 in UDT1002 can be recognized (basic program). In case of new alarm functions, the UDT1002 must be assigned symbolically to the DB2.

At the start of DB2 there are bit arrays for signals without alarm generation. This is followed by an array of size 64 integer values for additional info about the user area 0.

Thereafter follow the areas, which also trigger alarms / messages (see List manual) These areas are extended to 10 channels, 31 axes.

**Simple implementation of a user program on the new alarms**

The source container of the basic program contains the file "udt2_for_Convert.awl", which has the following structural element from UDT1002:

- ChanA as array of 1 ... 8
- AxisA as array of 1 ... 18
- UserA as array of 0 ... 63

This UDT2 is to be compiled via the LAD/FBD/STL editor. The UDT2 must be assigned to the DB2 in the symbol table.

Sources must be generated for components, which have assignments on DB2. Alternatively, sources can naturally be created for all blocks too. The UDT1002 must now be assigned to the DB2 in the symbol table. Thereafter, the sources must be recompiled.

Now all the alarm allocations are assigned to the new data areas in the DB2 and now only the parameter "ExtendAlMsg" at FB1 must be set to True.

After a Power On RESET the alarm behavior is the same as earlier.
13.9.3 PLC/MCP/HHU interface

General

There are different connection options for the machine control panel (MCP) and the handheld unit (HHU). This is in part due to the history of the MCP and HHU. This description focuses primarily on the connection of the Ethernet components.

On the SINUMERIK 840D sl, the machine control panel (MCP) and handheld unit (HHU) are connected via the Ethernet bus, which also links the TCU to the NCU. The advantage of this is that only one bus cable is required to connect the operating unit.

Topology SINUMERIK 840D sl

On the 840 D, the machine control panel and the handheld unit are connected to the CP 840D sl Ethernet bus (see Figure below). Where the connection of further keys and displays is required for customized operator panels, an additional keyboard interface (machine control panel without operating unit) can be used. For each keyboard interface, 64 pushbuttons, switches, etc. and 64 display elements can be connected via ribbon cable.

The signals sent from the MCP are copied to the PLC’s DPR (Dual-Port RAM) by the integrated Ethernet CP-840D sl. The basic program of the PLC enters the incoming signals in the input image configured at FB1. The NC-related signals are generally distributed by the basic program to the NC/PLC interface. If required, the signals can be modified by the user.

The signals from the PLC to the MCP (displays) are transferred in the opposite direction.
Bus addresses

On Ethernet components, MAC and IP addresses or logic names are determining factors in respect of communication. The control system's system programs convert logic names into MAC or IP addresses. On the PLC, the numeric component of the logic name is used for communication. This numeric part is specified by the user to the FB1 using parameter "MCPxBusAdr".

The logical name of an MCP or HHU always begins with "DIP". This is followed by a number corresponding to the switch position of the MCP component (e.g. DIP192, DIP17).

MCP interface in the PLC

The signals from the machine control panel are routed by default via the I/O interface to the PLC area. A distinction must be made between NC and machine-specific signals. NC-specific key signals are distributed to the relevant mode-group-, NCK-, axis- and spindle-specific interface by FC19 (or FC24, FC25, FC26, depending on the type of MCP) by default. The reverse applies to the associated status signals which are routed to the MCP interface. For this purpose, FC19 or the other blocks mentioned above must be called in the user program.

Customized keys, which can be used to trigger a wide range of machine functions, must be evaluated directly by the user program. The user program also routes the status signals to the output area for the LEDs.
13.10 Structure and functions of the basic program

General

The PLC program has a modular structure. The organization blocks (OB) form the interface between the operating system and the basic and user programs.

- Restart (warm restart) with start-up and synchronization (OB100)
- Cyclic operation (OB1)
- Process alarms (OB40)
- Asynchronous errors: Diagnostics alarm (OB82), module failure (OB86)

The calls of the function blocks of the basic and user programs must be programmed by the user in the organization blocks (OB).
13.10.1 Startup and synchronization of NCK PLC

Loading the basic program

The basic program must be loaded with the S7 tool when the PLC is in the Stop state. This ensures that all blocks in the basic program will be initiated correctly the next time they are called. Otherwise, undefined states can occur in the PLC (e.g. blinking of all PLC LEDs).

Startup,

The synchronization of NCK and PLC is performed during startup. The system and user data blocks are checked for integrity and the most important basic program parameters are verified for plausibility. In cases of errors, the basic program produces an alarm (visible on HMI) and switches the PLC to the Stop state.

A warm restart is not provided, i.e. following system initialization, the operating system runs organization block OB100 and always commences cyclic execution at the start of OB1.
Synchronization

The PLC is synchronized with the HMI and NCK and CP during powerup.

Sign-of-life

After a correct initial start and the first complete OB1 cycle (initial setting cycle) the PLC and NCK continuously exchange sign-of-life signals. If the sign of life from the NCK fails to materialize, the PLC/NCK interface is initialized and the signal "NCK CPU ready" in DB10 is set to FALSE.

13.10.2 Cyclic operation (OB1)

General

The NCK PLC interface is processed completely in cyclic mode. From a chronological viewpoint, the basic program runs ahead of the user program. In order to minimize the execution time of the basic program, only the control/status signals are transmitted cyclically; transfer of the auxiliary functions and G functions only takes place on request.

The following functions are performed in the cyclic part of the basic program:

- Transmission of the control/status signals
- Distribution of the auxiliary functions
- M decoding (M00 - M99),
- M, S, F distribution
- Transfer the MCP signals via NCK
- Acquisition and conditioning of the user errors and operating messages.

Control/status signals

A shared feature of the control and status signals is that they are bit fields. The basic program updates them at the start of OB1.

The signals can be subdivided into the following groups:

- General signals
- Mode group-specific signals (such as mode types)
- Channel-specific signals (such as program and feed modifications)
- Axis- and spindle-specific signals (such as feed disable)
Auxiliary and G functions

The auxiliary and G functions have the following characteristics:

- Transfer to the PLC is block-synchronous (referred to a part program block)
- Transfer is acknowledge-driven.
- The acknowledgement times have an immediate effect on the execution time of NC blocks containing auxiliary functions requiring acknowledgement.

The value range is presented in the table below:

<table>
<thead>
<tr>
<th>Function</th>
<th>Structure</th>
<th>Value range</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>G function</td>
<td>G function</td>
<td>255(1)</td>
<td>Byte</td>
</tr>
<tr>
<td>M word</td>
<td>M group M word</td>
<td>99 99.999.999</td>
<td>Word DWord</td>
</tr>
<tr>
<td>S word</td>
<td>Spindle no. S word</td>
<td>6 Floating point(2)</td>
<td>Word DWord</td>
</tr>
<tr>
<td>T word</td>
<td>Magazine no. T word</td>
<td>99 65535</td>
<td>Word Word</td>
</tr>
<tr>
<td>D word</td>
<td>- D word</td>
<td>99 255</td>
<td>Byte Byte</td>
</tr>
<tr>
<td>H word</td>
<td>H group H word</td>
<td>99 Floating point</td>
<td>Word DWord</td>
</tr>
<tr>
<td>F word</td>
<td>Axis no. F word</td>
<td>18 Floating point</td>
<td>Word DWord</td>
</tr>
</tbody>
</table>

1) relative number, transferred for each G group
2) corresponding STEP 7 format (24-bit mantissa, 8-bit exponent)

The M, S, T, H, D and F values sent by the NCK are output together with the accompanying change signals to the CHANNEL DB interface via the auxiliary/G functions (see List Manual). The function value and the extended address are transferred to the appropriate data word. The accompanying modification signal is activated to 1 for one PLC cycle. When the modification signal is reset, the acknowledgement is passed to the NCK. The acknowledgement of high-speed auxiliary functions is given by the basic program immediately the basic program detects the auxiliary function.

In addition to distribution of the auxiliary and G functions, selected signals are processed as described below.

M decoder

M functions can be used to transfer switching commands and fixed-point values. Decoded dynamic signals are output to the CHANNEL DB interface for standard M functions (range M00 - M99) signal length = 1 cycle time).
G group decoders

In the case of G functions sent by the NCK, the related groups are decoded and the current G number is entered in the corresponding interface byte of the CHANNEL DB, i.e. all active G functions are entered in the channel DBs. The entered G functions are retained even after the NC program has terminated or aborted.

Note
During system startup, all G group bytes are initialized with the value "0".

M, S, F distributor

The M, S, F, distributor is used to enter spindle-specific M words M(1...6)=[3,4,5], S words and F words for axial feeds in the appropriate spindle and axis data blocks. The criterion for distribution is the extended address which is passed to the PLC for M words, S words and axial F words.

MCP signal transmission

Depending on the bus connection, the MCP signals are either transmitted directly to the PLC or indirectly to the parameterized I/O areas via an internal procedure using the basic program.

User messages

The acquisition and processing of the user error and operational messages is performed by an FC in the basic program.

13.10.3 Time-interrupt processing (OB35)

The user must program OB35 for time-interrupt processing. The default time base setting of OB 35 is 100 ms. A different time base can be selected using the STEP7 "HW Config" tools. However, the OB35 time setting must be at least 3 ms in order to avoid a PLC CPU stop. The stop is caused by reading of the HMI system state list during powerup of the HMI. This reading process blocks priority class control for approx. 2 ms. The OB35 with a time base set to a rather lower value is then no longer processed correctly.

13.10.4 Process-interrupt processing (OB 40)

A process interrupt OB40 (interrupt) can, for example, be triggered by appropriately configured I/Os or by certain NC functions. Due to the different origin of the interrupt, the PLC user program must first interpret the cause of the interrupt in OB40. The cause of the interrupt is contained in the local data of OB40.

References:
SIMATIC STEP 7 Description or Online Help of STEP 7
13.10.5 Diagnostic alarm, module failure processing (OB82, OB86)

General

In the event of a diagnostic alarm or failure of an I/O module, organization block OB82 or OB86 is called (components of the basic program). The basic program block FC5 "Diagnostic alarm and module failure" (Page 994) is called from these blocks.

Bus diagnostics

The DP slave modules of the PROFIBUS connections MPI/DP or DP1 are registered by the basic program as group message in the form of a ready signal in the interface signal

- DB10.DBX92.0 (MPI/DP bus slaves OK)
- DB10.DBX92.1 (DP1 bus slaves OK)
- DB10.DBX92.2 (PN bus slaves OK)

The group message is derived from the LED status of the respective PROFIBUS (System state list SZL 0x174).

Alarm Output

If a fault or a failure of a slave is detected, alarm 400551, 400552 or 400553 (fault on the MPI/DP, DP or PN bus) is also signaled. The alarm is automatically deleted when the error is removed.

Suppression of the alarm output 400551, 400552, 400553

By setting the individual bits to "TRUE", the output of the group fault message on the HMI is suppressed:

- DB10.DBX92.4 (alarm suppression message 400551) = TRUE
- DB10.DBX92.5 (alarm suppression message 400552) = TRUE
- DB10.DBX92.6 (alarm suppression message 400553) = TRUE

It is irrelevant whether the bit is set before the error occurs or during and active error.

If removed (reset of the bit to "FALSE"), the appropriate message is displayed on the HMI in the event of an error.

In addition, bits DBX92.4-6 are reset to "FALSE" during the control power-up, so that the evaluation of the error messages is always activated after a control restart.

Note

By setting the bit, the fault monitoring for the complete bus line is deactivated!
13.10.6 Response to NCK failure

General

During cyclic operation, the PLC basic program continuously monitors NCK availability by polling the signoflife character. If the NCK is no longer reacting, the NCK PLC interface is initialized, and the NCK CPU ready signal in the area of the signals from NC (DB 10.DBX 104.7) is reset. Furthermore, the signals sent from the NCK to the PLC and vice versa are set to an initial state.

The PLC itself remains active so that it can continue to control machine functions. However, it remains the responsibility of the user program to set the machine to a safe state.

NCK → PLC signals

The signals sent by the NCK to the PLC are divided into the following groups:

- Status signals from the NCK, channels, axes and spindles
- Modification signals of the auxiliary functions
- Values of the auxiliary functions
- Values of the G functions

Status signals:
The status signals from the NCK, channels, axes, and spindles are reset.

Auxiliary-function modification signals:
Auxiliary-function modification signals are also reset.

Auxiliary-function values:
Auxiliary-function values are retained so that it is possible to trace the last functions triggered by the NCK.

G-function values:
G function values are reset (i.e. initialized with the value 0 respectively).

PLC → NCK signals

The signals sent by the PLC to the NCK are divided into control signals and tasks that are transferred by FCs to the NCK.

Control signals:
The control signals from the PLC to the NCK are frozen; cyclic updating by the basic program is suspended.

Jobs from PLC to NCK:
The FCs and FBs, which are used to pass jobs to the NCK, must no longer be processed by the PLC user program, as this could lead to incorrect checkback signals. During powerup of the control, a job (e.g. read NCK data) must not be activated in the user program until the NCK CPU ready signal is set.
13.10.7 Functions of the basic program called from the user program

General

In addition to the modules of the basic program, which are called at the start of OB1, OB40 and OB100, functions are also provided which have to be called and supplied with parameters at a suitable point in the user program.

These functions can be used, for example, to pass the following jobs from the PLC to the NCK:

- Traversing concurrent axes (FC18)
- Start asynchronous subprograms (ASUBs) (FC9),
- Select NC programs (FB4)
- Control spindle (FC18),
- Read and write variables (FB2, FB3)

Note

Checking and diagnostics of a function call of the basic PLC program

To simplify the checking and diagnostics of a function call (FB or FC) of the basic PLC program that is controlled via a trigger (e.g. via Req, Start parameters) and that provide an execution acknowledgement as output parameter (e.g. via Done, NDR, Error parameters), proceed as follows.

A variable compiled of other signals which produce the trigger for the function call should be set. Start conditions may be reset only as a function of the states of parameters Done, NDR and Error.

The appropriate control mechanism can be placed in front of or behind the function call. If the mechanism is placed after the call, the output variables can be defined as local variables (advantage: Reduction of global variables, markers, data variables and time-related advantages over data variables).

The trigger parameter must be a global variable (e.g. marker, data variable).

Jobs that are still active must be reset from the user program in OB100 (Req, Start, parameters, etc. from TRUE ⇒ FALSE). A POWER OFF/ON could result in a state in which jobs are still active.

Concurrent axes

The distinguishing features of concurrent axes are as follows:

- They must be defined as such via the NC machine data.
- They can be traversed either from the PLC or from the NC by means of the JOG keys.
- Starting from the PLC is possible in the NC operating modes MDI and AUTOMATIK via FC.
- The start is independent of NC block boundaries.

Function calls are available for positioning axes, indexing axes and spindles (FC18).
Asynchronous subprograms (ASUBs)

An ASUB can be used to activate arbitrary functions in the NCK. Before an asynchronous subprogram can be started from the PLC, it must be ensured that it is available and prepared by the NC program or by FB4 PI services (ASUB).

Once prepared in this way, it can be started at any time from the PLC. The NC program running in one of the parameterized channels of FC9 is interrupted by the asynchronous subprogram. An ASUB is started by calling FC9 from the user program by setting the start parameter to 1.

Note

If an asynchronous subprogram has not been prepared by an NC program or by FB4 (ASUB) (e.g. if no interrupt no. has been assigned), a start error is output (StartErr = TRUE).

Read/Write NC variables

NCK variables can be read with FB GET while values can be entered in NCK variables with FB PUT. The NCK variables are addressed via identifiers at inputs Addr1 to Addr8. The identifiers (symbols) point to address data which must be stored in a global DB. To allow generation of this DB, the PC software is supplied with the basic program with which the required variables can be selected from a table, which is also supplied. The selected variables are first collected in a second, project-related list. Command Generate DB creates a "*.AWL" file which must be linked to the program file for the machine concerned and compiled together with the machine program.

1 to 8 values can be read or written with a read or write job. If necessary, the values are converted (e.g. NCK floating-point values (64-bit) are converted to PLC format (32-bit with 24-
A loss of accuracy results from the conversion from 64-bit to 32-bit REAL. The maximum precision of 32-bit REAL numbers is approximately $10^7$.

AG_SEND/AG_RECV functions

The AG_SEND/AG_RECV functions correspond to the functions of the library "SIMATIC_NET_CP" of the S7-300 CPU in STEP 7. In general, the online help is valid for these functions.

The AG_SEND/AG_RECV functions can be used for data exchange with another station via the integrated "CP 840D sl". A description of the functions is provided in Chapter "Block descriptions (Page 914)".

Note

Other communication blocks (e.g. BSEND, USEND) which possess a CP343-1 are not supported in SINUMERIK 840D sl.

13.10.8 Symbolic programming of user program with interface DB

General

Note

The basic program library on the CD supplied with the Toolbox for the 840D contains files NST_UDTB.AWL and TM_UDTB.AWL.

The compiled UDT blocks from these two files are stored in the CPU program of the basic program.

A UDT is a data type defined by the user that can, for example, be assigned to a data block generated in the CPU.

Symbolic names of virtually all the interface signals are defined in these UDT blocks.

The UDT numbers 2, 10, 11, 19, 21, 31, 71, 72, 73, 77, 1002, 1071, 1072, 1073 are used.
The assignments have been made as follows:

<table>
<thead>
<tr>
<th>UDT number</th>
<th>Assignment to interface DB</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDT2</td>
<td>DB2</td>
<td>Alarms/messages</td>
</tr>
<tr>
<td>UDT10</td>
<td>DB10</td>
<td>NCK signals</td>
</tr>
<tr>
<td>UDT11</td>
<td>DB11</td>
<td>Mode group signals</td>
</tr>
<tr>
<td>UDT19</td>
<td>DB19</td>
<td>HMI signals</td>
</tr>
<tr>
<td>UDT21</td>
<td>DB21 to DB30</td>
<td>Channel signal</td>
</tr>
<tr>
<td>UDT31</td>
<td>DB31 to 61</td>
<td>Axis/spindle signals</td>
</tr>
<tr>
<td>UDT71</td>
<td>DB71</td>
<td>Tool management: Load/unload locations</td>
</tr>
<tr>
<td>UDT72</td>
<td>DB72</td>
<td>Tool management: Change in spindle</td>
</tr>
<tr>
<td>UDT73</td>
<td>DB73</td>
<td>Tool management: Change in revolver</td>
</tr>
<tr>
<td>UDT77</td>
<td>DB77</td>
<td>MCP and HHU signals with standard SDB 210</td>
</tr>
<tr>
<td>UDT1002</td>
<td>DB2</td>
<td>Extended alarms / messages (FB1 parameter “ExtAlMsg:=TRUE”)</td>
</tr>
<tr>
<td>UDT 1071</td>
<td>DB 1071</td>
<td>Tool management: Loading/unloading points (multitool)</td>
</tr>
<tr>
<td>UDT 1072</td>
<td>DB 1072</td>
<td>Tool management: Change in spindle (multitool)</td>
</tr>
<tr>
<td>UDT 1073</td>
<td>DB 1073</td>
<td>Tool management: Change in turret (multitool)</td>
</tr>
</tbody>
</table>

To symbolically program the interface signals, the interface data blocks must first be symbolically assigned using the symbol editor.

For example, symbol "AxisX" is assigned to operand DB31 with data type UDT31 in the symbol file.

After this input, the STEP 7 program can be programmed in symbols for this interface.

**Note**

Programs generated with an earlier software version that utilize the interface DBs described above can also be converted into symbol programs. A fully qualified command for data access e.g. "U DB31.DBX60.0" is then necessary for (spindle / no axis) in the program created till now. This command is converted upon activation of the symbolics in the editor "AxisX.E_SpKA".

**Description**

Abbreviated symbolic names of the interface signals are defined in the two STL files NST_UDTB.AWL and TM_UDTB.AWL.

In order to create the reference to the names of the interface signals, the name is included in the comment after each signal.

The names are based on the English language. The comments are in English.
The symbolic names, commands and absolute addresses can be viewed by means of a STEP 7 editor command when the UDT block is opened.

### Note

Unused bits and bytes are listed, for example, with the designation "f56_3".

- "56": Byte address of the relevant data block
- "3": Bit number in this byte

#### 13.10.9 M decoding acc. to list

**Function description**

Up to 256 M functions with extended address can be decoded from the basic program using the "M decoding according to list" function. The function is activated using FB1 parameter "ListMDecGrp" (number of M groups for decoding). The assignment of the M function with extended address and a signal in the signal list is defined in the decoding list. The signals are also grouped for this purpose.

**Decoding list (DB75)**

The source file for the decoding list (MDECLIST.AWL) is supplied with the basic program. Data block DB75 is created when the STL source is compiled. Before the function is activated, the decoding list (DB75) must be transferred to the PLC followed by a restart.

An M function is decoded if it is in the decoding list. When decoding the M function, the corresponding signal is set in the signal list as a function of the specific group. When setting a signal in the signal list, the interface signal "Read in inhibit" is set by the basic program in the associated channel of the NC. The interface signal is reset again for the channel as soon as the user resets all of the signals output from this channel in the signal list; i.e. after they have been acknowledged.

**Signal list (DB76)**

When activating the function in data block DB76, the basic program creates the signal list. From then, for each M signal decoded according to the list, a signal is set in the signal list (DB76) in the corresponding group. At the same time, the "Readin inhibit" interface signal is set in the channel in which the M function has been output. The interface signal is reset again for the channel as soon as the user resets all of the signals output from this channel in the signal list; i.e. after they have been acknowledged.

**Highspeed auxiliary functions**

When an M function contained in the decoding list is output as "fast help function", no read-in inhibit is set for the corresponding channel of the NC.

The figure below shows the structure of the **M decoding according to list**:
Activation

M decoding is activated using FB1 parameter "ListMDecGrp"

The number of M groups to the evaluated and/or decoded is specified using the appropriate parameter. The function is active for a parameter value = 1 ... 16.

- Basic program, OB100, FB1 parameter ListMDecGrp = <number of M groups> (also see " FB1: RUN_UP - basic program, start section (Page 914) ").

Properties and structure of the decoding list (DB75)

Properties of the decoding list (DB75):

- There is only one decoding list independent of the channel.
- The decoding list can include a maximum of 16 groups.
- A group has a maximum of 16 signals
• There must be an entry in the decoding list for every group of M functions to be decoded.

• The assignment between the M function with extended address and the signal to be set in the signal list is specified in the decoding list using the first and last M function of the associated group.
  – First M function: Parameter: "MFirstAdr" ≙ signal or bit 0
  – Last M function: Parameter: "MLastAdr" ≙ dependent on the difference to the first M function maximum signal or bit 15

Structure of the decoding list (DB75):

An entry in the decoding lists consists of 3 parameters, each of which is assigned to a group.

<table>
<thead>
<tr>
<th>Group</th>
<th>Extended M address</th>
<th>First M address of the group</th>
<th>Last M address of the group</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MSigGrp[1].MExtAdr</td>
<td>MSigGrp[1].MFirstAdr</td>
<td>MSigGrp[1].MLastAdr</td>
</tr>
<tr>
<td>2</td>
<td>MSigGrp[2].MExtAdr</td>
<td>MSigGrp[2].MFirstAdr</td>
<td>MSigGrp[2].MLastAdr</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>16</td>
<td>MSigGrp[16].MExtAdr</td>
<td>MSigGrp[16].MFirstAdr</td>
<td>MSigGrp[16].MLastAdr</td>
</tr>
</tbody>
</table>

Type and value range of the signals:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MExtAdr</td>
<td>INT</td>
<td>0 ... 99</td>
<td>Extended M address</td>
</tr>
<tr>
<td>MFirstAdr</td>
<td>DINT</td>
<td>0 to 99.999.999</td>
<td>First M address in group</td>
</tr>
<tr>
<td>MLastAdr</td>
<td>DINT</td>
<td>0 to 99.999.999</td>
<td>Last M address in group</td>
</tr>
</tbody>
</table>

Properties of the signal list (DB76)

The signal list (DB76) has the following properties:

• There is only one signal list independent of the channel.
• The signal list can include a maximum of 16 signals for each M group.

Example

3 groups of M functions are to be decoded:

• Group 1: M2 = 1 to M2 = 5
• Group 2: M3 = 12 to M3 = 23
• Group 3: M40 = 55

Structure of the decoding and signal list

<table>
<thead>
<tr>
<th>Group</th>
<th>Decoding list (DB75)</th>
<th>Signal list (DB76)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Extended M address</td>
<td>First M address</td>
</tr>
<tr>
<td></td>
<td>of the group</td>
<td>of the group</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>55</td>
</tr>
</tbody>
</table>
Program code
DATA_BLOCK DB 75
TITLE =
VERSION : 0.0
STRUCT
   MSigGrp : ARRAY [1 .. 16 ] OF STRUCT
       MExtAdr : INT;
       MFirstAdr : DINT;
       MLastAdr : DINT;
   END_STRUCT ;
BEGIN
   MSigGrp[1].MExtAdr   :=    2; extended M address of the 1st group
   MSigGrp[1].MFirstAdr := L#1; first M address of the group
   MSigGrp[1].MLastAdr  :=  L#5; last M address of the group
   MSigGrp[2].MExtAdr   :=    3; extended M address of the 2nd group
   MSigGrp[2].MFirstAdr := L#12; first M address of the group
   MSigGrp[2].MLastAdr  := L#23; last M address of the group
   MSigGrp[3].MExtAdr   :=   40; extended M address of the 3rd group
   MSigGrp[3].MFirstAdr := L#55; first M address of the group
   MSigGrp[3].MLastAdr  := L#55; last M address of the group
END_DATA_BLOCK

Structure of FB1 in OB100
To activate the function, insert the parameter for the number of M groups to be decoded "ListMDecGrp".

Call FB 1, DB 7(
   ...
   ListMDecGrp := 3;  //M decoding of three groups
   ...
);

Description
A restart must be performed after the entry has been made in OB100 and the decoding list (DB75) has been transferred to the PLC. The basic program creates the signal list (DB76) when it restarts.

An NC program is then started, for instance in the 1st channel. An extended M function is included in this (M3=17). When decoding the M function (M3 ≡ group 2), the associated signal (DBW1.5) is set in the signal list (DB76) and the interface signal "Read-inhibit" in the 1st channel. The execution of the NC program is stopped. Further, the "Extended address M function" and the "M function number" are displayed in the channel DB of the 1st channel.

The "Read-in inhibit" signal in the 1st channel is reset once the user has reset all of the signals output from this channel in the signal list (DB76), and has therefore acknowledged them.
13.10.10 PLC machine data

General

The user has the option of storing PLC-specific machine data in the NCK. These machine data can then be processed during power-up of the PLC (OB100). This enables, for example, user options, machine expansion levels, machine configurations, etc., to be implemented.

The interface to read this data is in DB20. However, DB20 is set up by the basic program during powerup only when user machine data are used, i.e., sum of GP parameters "UDInt", "UDHex" and "UDReal" is greater than ZERO.

The sizes of the individual areas, and hence the total length of the DB20, is set by the following PLC machine data:

- MD14504 $MN_MAXNUM_USER_DATA_INT
- MD14506 $MN_MAXNUM_USER_DATA_HEX
- MD14508 $MN_MAXNUM_USER_DATA_FLOAT

These settings are specified to the user in the GP parameters "UDInt", "UDHex" and "UDReal".

The data is stored in the DB20 by the GP seamlessly in the sequence:

1. Integer MD
2. Hexa-fields MD
3. Real MD

The integer and real values are stored in DB20 in S7 format.

Hexadecimal data is stored in DB20 in the order in which they are input (use as bit arrays).

Figure 13-13 DB20
Note

If the number of PLC machine data used is increased later, then DB20 must be deleted beforehand. To prevent such extensions in use having any effect on the existing user program, the data in DB20 should be accessed in symbolic form wherever possible, e.g. by means of a structure definition in the UDT.

<table>
<thead>
<tr>
<th>Alarms</th>
<th>Deleting DB20 in the PLC and restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation</td>
<td>DB length is not the same as the required DB length</td>
</tr>
<tr>
<td>Reaction</td>
<td>Alarm display and PLC stop</td>
</tr>
<tr>
<td>Remedy</td>
<td>Delete DB20 followed by a RESET</td>
</tr>
<tr>
<td>Continuation</td>
<td>After cold restart</td>
</tr>
</tbody>
</table>

Example

The project in the example requires 4 integer values, 2 hexadecimal fields with bit information and 1 real value.

Machine data:

```
MD14510 $MN_USER_DATA_INT[0] 123
MD14510 $MN_USER_DATA_INT[1] 456
MD14510 $MN_USER_DATA_INT[2] 789
MD14510 $MN_USER_DATA_INT[3] 1011
...
MD14512 $MN_USER_DATA_HEX[0] 12
MD14512 $MN_USER_DATA_HEX[1] AC
...
MD14514 $MN_USER_DATA_FLOAT[0] 123.456
```

GP parameter (OB100):

```
CALL FB1, DB7(
    MCPNum := 1,
    MCP1In := P#E0.0,
    MCP1Out := P#A0.0,
    MCP1StatSend := P#A8.0,
    MCP1StatRec := P#A12.0,
    MCP1BusAdr := 6,
    MCP1Timeout := S5T#700MS,
    MCP1Cycl := S5T#200MS,
    NCCyclTimeout := S5T#200MS,
    NCRunupTimeout := S5T#50S;
)```

Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2

865
BP parameters (to scan runtime):

\[
\begin{align*}
  &1 \text{ gp\_par.UDInt; } //=4, \\
  &1 \text{ gp\_par.UDHex; } //=2, \\
  &1 \text{ gp\_par.UDReal; } //=1 \\
\end{align*}
\]

During PLC power-up, DB20 was generated with a length of 28 bytes:

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>123</td>
</tr>
<tr>
<td>2.0</td>
<td>456</td>
</tr>
<tr>
<td>4.0</td>
<td>789</td>
</tr>
<tr>
<td>6.0</td>
<td>1011</td>
</tr>
<tr>
<td>8.0</td>
<td>b#16#12</td>
</tr>
<tr>
<td>9.0</td>
<td>b#16#AC</td>
</tr>
<tr>
<td>10.0</td>
<td>1.234560e+02</td>
</tr>
</tbody>
</table>

The structure of the machine data used is specified in a UDT:

```
TYPE UDT20
  STRUCT
    UDInt : ARRAY [0 .. 3] OF INT;
    UDHex0 : ARRAY [0 .. 15] OF BOOL;
    UDReal : ARRAY [0 .. 0] OF REAL;  //Description as field, for
    // later expansions
  END_STRUCT;
END_TYPE
```

Note

ARRAY OF BOOL are always sent to even-numbered addresses. For this reason, an array range of 0 to 15 must generally be selected in the UDT definition or all Boolean variables specified individually.

Although only a REAL value is used initially in the example, a field (with one element) has been created for the variable. This ensures that extensions can be made easily in the future without the symbolic address being modified.

Symbolic accesses

An entry is made in the symbol table to allow data access in symbolic form:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operand</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>UData</td>
<td>DB20</td>
<td>UDT20</td>
</tr>
</tbody>
</table>
Access operations in user program (list includes only symbolic read access):

```
L  "UData".UDInt[0];
L  "UData".UDInt[1];
L  "UData".UDInt[2];
L  "UData".UDInt[3];
U  "UData".UDHex0[0];
U  "UData".UDHex0[1];
U  "UData".UDHex0[2];
U  "UData".UDHex0[3];
U  "UData".UDHex0[4];
U  "UData".UDHex0[5];
U  "UData".UDHex0[6];
U  "UData".UDHex0[7];
U ...;
U  "UData".UDHex0[15];
L  "UData".UDReal[0];
...
```

13.10.11 Configuration machine control panel, handheld unit, direct keys

General

Up to two machine control panels and one handheld unit can be in operation at the same time. There are various connection options (Ethernet/PROFINET, PROFIBUS) for the machine control panel (MCP) and handheld unit (HHU). It is possible to connect two MCPs to different bus systems (mixed operation is only possible on Ethernet and PROFIBUS). This can be achieved using FB1 parameter "MCPBusType". In this parameter, the right-hand decade (units position) is responsible for the first MCP and the left-hand decade (tens position) for the second MCP.

Parameterization of components is always performed by calling the basic program block FB1 in OB100. FB1 saves its parameters in the associated instance data block (DB7, symbolic "gp_par"). Separate parameter sets are provided for each machine control panel and the handheld unit. The input/output addresses of the user must be defined in these parameter sets. These input and output addresses are also used in FC19, FC24, FC25, FC26 and FC13. Further, the addresses for status information, PROFIBUS or Ethernet/PROFINET are also to be defined. The default time settings for timeout and cyclic forced retriggering should not be changed. Please refer to the Operator Components manual for further information on MCPs and handheld unit components.
Activation

Each component is activated either via the number of machine control panels ("MCPNum" parameter) or, in the case of the handheld unit, via the "HHU" parameter. The MCP and HHU connection settings are entered in FB1 parameters "MCPMPI", "MCPBusType" or "BHGMPI".

Handheld unit (HT 2)

In the handheld unit the addressing is done via a parameter of the GD parameter set. This was necessary for reasons of compatibility of the parameter names.

Configuration

Essentially, there are various communication mechanisms for transferring data between the MCP/HHU and PLC. These mechanisms are characterized by the bus connection of the MCP and HHU. In one case (Ethernet), data is transported via the "CP 840D sl".

The mechanism is parameterized completely via the MCP/HHU parameters in FB1.

In the other case the transmission is via the PLC operating system through the PROFIBUS configuration.

The parameter setting is done via STEP 7 in HW-Config. To enable the basic program to access the data and failure monitoring of MCP/HHU, the addresses set in FB1 parameters must be made available to the basic program.

An overview of the various coupling mechanisms is shown below. Mixed operation can also be configured.

If an error is detected due to a timeout monitor, an entry is made in the alarm buffer of the PLC CPU (alarms 400260 to 400262). In this case, the input signals from the MCP or from the handheld unit (MCP1In/MCP2In or BHGIn) are reset to 0. If it is possible to resynchronize the PLC and MCP/HHU, communication is resumed automatically and the error message reset by the GP.

Note

The abbreviation "(n.r.)" in the tables below means "Not relevant".

Ethernet connection (MCPBusType = 5)

Without further configuration settings being made, communication takes place directly from the PLC GP via the CP 840D sl. The FB1 parameters listed below are used for parameterization.

The numeric part of the logical name of the component must be entered in "MCP1 BusAdr", "MCP2 BusAdr" or "BHGREcGDNo" (corresponds to the bus address of the node). The logical name is defined via switches on the MCP or terminal box.
An error entry is also made in the PLC alarm buffer for timeouts. As a result, the following error messages are output at the HMI:

- 400260: MCP 1 failure
  or
- 400261: MCP 2 failure
- 400262: HHU failure

An MCP or HHU failure is detected immediately after a cold restart even if no data has yet been exchanged between the MCP/HHU and PLC.

The monitoring function is activated as soon as all components have signaled "Ready" after powerup.
Example: OP with direct keys

The direct keys of the OPs at the Ethernet bus should be transferred to the PLC. Previously, the direct keys have been transferred to the PLC via the PROFIBUS or via a special cable connection between OP and MCP.

For connecting the direct keys via the Ethernet, this concerns e.g. the "OP 08T", there is a parameterization in the basic program for activating the data transport. The related parameters lie in the instance DB of the FB 1 (OpKeyNum to OpKeyBusType, see data table). The parameters are provided by the user in the start OB100 by connecting the parameter at the FB1 call. The bus address and Op1/2KeyStop can also be modified in the cyclic operation by writing the FB1 instance DB DB 7.

The transport of the user data of the direct keys runs in the same way as in the case of Ethernet MCP. The data transport can also be stopped and restarted via writing the DB7-parameter "Op1/2KeyStop". During the Stop phase the address of the direct key module (TCU-index or the MCP-address) can also be changed.

After resetting the Stop signal, a connection to the new address is established.

The status of the respective direct-key interfaces can be read in the interface signal:

DB10.DBX104.3 (OP1Key ready)

or

DB10.DBX104.4 (OP2Key ready)

Address direct keys

For the parameter Op1/2KeyBusAdr the TCU index is normally to be used. This affects the OPs, such as OP 08T, OP 12T, which for the direct keys do not have special cable connection to an Ethernet MCP.

If OPs with direct keys have a special cable connection and these are connected to an Ethernet-MCP, then for the parameter Op1/2KeyBusAdr the address of the MCP (DIP-switch setting of the MCP) is to be used. Only the data stream of the direct keys (2 bytes) is transferred via the direct key interface.

Alarm direct keys

An error entry is also made in the PLC alarm buffer for timeouts. As a result, the following error messages are output at the HMI:

- 400274: Direct key 1 failed

  or

- 400275: Direct key 2 failed

Control unit switching for direct keys

The user connects Op1/2KeyBusAdr with 0xFF and stop = TRUE in start block OB100. The direct key address of the M-to-N interface is connected to parameter "Op1KeyBusAdr" via the M-to-N block FB9.

<table>
<thead>
<tr>
<th>Relevant parameters (FB1)</th>
<th>e.g. direct keys OP 08T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct keys</td>
<td></td>
</tr>
<tr>
<td>OpKeyNum = 1 or 2 (number of OPs with direct keys)</td>
<td></td>
</tr>
<tr>
<td>Op1KeyIn</td>
<td>Op2KeyIn</td>
</tr>
</tbody>
</table>
MCP identification

Via the identify interface in DB7 it is possible to query the type of the Ethernet component (MCP, HT 2, HT 8 or direct keys) with the relevant parameters at the input/output in cyclic operation:

- Relevant parameters at the input:
  "IdentMcpBusAdr", "IdentMcpProfilNo", "IdentMcpBusType", "IdentMcpStrobe"

- Relevant parameters at the output:
  "IdentMcpType", "IdentMcpLengthIn", "IdentMcpLengthOut"

Here the DIP device address or the TCU index at the parameter "IdentMcpBusAdr" is activated by the user program together with setting of the Strobe signal.

The input parameter "IdentMcpProfilNo" is normally to be set to the value 0. This parameter is to be set to the value 1 only in the identification of the direct keys. The parameter "IdentMcpBusType" currently has no significance for a user program and is to be left in its default value.

After resetting the Strobe signal by the basic program, valid output information becomes available to the user. The resetting of the Strobe signals by the basic program can last for several PLC cycles (up to two seconds).

The output parameters should show the user the size of the data areas for the addressed device. Furthermore, it can be defined here whether an HT 2 or an HT 8 or no device is connected to the connection box. With this information, the MCP channel or the HHU channel can be activated. In the cyclic operation the parameters can be written symbolically by the user program and read via the symbol names of the DB7 (gp_par).
### Relevant parameters (FB1)

<table>
<thead>
<tr>
<th>MCP device identification</th>
<th>Input parameters, e.g. OP 08T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct keys such as e.g. OP 08T, OP 12T</td>
<td>B#16#1</td>
</tr>
<tr>
<td><strong>IdentMcpType (Mcp-Type)</strong></td>
<td></td>
</tr>
<tr>
<td>no device connected</td>
<td>0</td>
</tr>
<tr>
<td>MCP 483C IE (Compact)</td>
<td>B#16#80</td>
</tr>
<tr>
<td>MCP 483C IE</td>
<td>B#16#81</td>
</tr>
<tr>
<td>MCP 310</td>
<td>B#16#82</td>
</tr>
<tr>
<td>MCP OEM</td>
<td>B#16#83</td>
</tr>
<tr>
<td>MCP DMG</td>
<td>B#16#84</td>
</tr>
<tr>
<td>HT 8</td>
<td>B#16#85</td>
</tr>
<tr>
<td>TCU_DT (direct keys)</td>
<td>B#16#86</td>
</tr>
<tr>
<td>MCP_MPP</td>
<td>B#16#87</td>
</tr>
<tr>
<td>HT 2</td>
<td>B#16#88</td>
</tr>
<tr>
<td>OP 08T (direct keys)</td>
<td>B#16#89</td>
</tr>
</tbody>
</table>

### PROFIBUS connection on the DP port (MCPBusType = 3)

In case of PROFIBUS connection of the MCP, this component must be considered in the hardware configuration setting of STEP 7. The MCP is connected to the standard DP bus of the PLC (not to MPI/DP). The addresses must be stored in the input and output log range. These start addresses must also be stored in the pointer parameters of the FB1. The FB1 parameters listed below are used for further parameterization.

There is no PROFIBUS version of the HHU. For this reason, an Ethernet connection is shown for the HHU in this figure. The PROFIBUS slave address must be stored in the parameters "MCP1BusAdr" and "MCP2BusAdr". Enter the pointer to the configured diagnostic address (e.g. P#A8190.0) in "MCPxStatRec".
**13.10 Structure and functions of the basic program**

### Relevant parameters (FB1)

<table>
<thead>
<tr>
<th>MCP</th>
<th>HHU</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCPNum = 1 or 2 (number of MCPs)</td>
<td>HHU = 5 (via CP 840D sl)</td>
</tr>
<tr>
<td>MCP1In</td>
<td>MCP2In</td>
</tr>
<tr>
<td>MCP1Out</td>
<td>MCP2Out</td>
</tr>
<tr>
<td>MCP1StatSend (n.r.)</td>
<td>MCP2StatSend (n.r.)</td>
</tr>
<tr>
<td>MCP1StatRec</td>
<td>MCP2StatRec</td>
</tr>
<tr>
<td>MCP1BusAdr</td>
<td>MCP2BusAdr</td>
</tr>
<tr>
<td>MCP1Timeout</td>
<td>MCP2Timeout</td>
</tr>
<tr>
<td>MCP1Cycl (n.r.)</td>
<td>MCP2Cycl (n.r.)</td>
</tr>
<tr>
<td>MCPMPI = FALSE</td>
<td>BHGCycl (n.r.)</td>
</tr>
<tr>
<td>MCP1Stop</td>
<td>MCP2Stop</td>
</tr>
<tr>
<td>MCPBusType = b#16#33</td>
<td>BHGRecObjNo (n.r.)</td>
</tr>
<tr>
<td>MCPSDB210 = FALSE</td>
<td>BHGSendGDNo (n.r.)</td>
</tr>
<tr>
<td>MCPCopyDB77 = FALSE</td>
<td>BHGSendGBZNo (n.r.)</td>
</tr>
<tr>
<td></td>
<td>BHGSendObjNo (n.r.)</td>
</tr>
</tbody>
</table>

MCP failure normally switches the PLC to the STOP state. If this is undesirable, OB 82, OB 86 can be used to avoid a stop. The basic program has, as standard, the OB82 and OB86 call. FC5 is called in these OBs. This FC5 checks whether the failed slave is an MCP. If this is the case, no PLC stop is triggered. Setting "MCPxStop" := TRUE causes the basic program to deactivate the MCP as a slave via SFC12. If the PLC does not switch to the stop state following the failure or fault of the MCP, an alarm message will be generated via the basic program. The interrupt is deleted when the station recovers.
PROFIBUS connection on the MPI/DP port (MCPBusType = 4)

With the PROFIBUS connection of the MCP, this component must be considered in the STEP 7 hardware configuration. The MCP is connected on the MPI/DP bus of the PLC.

The addresses must be stored in the input and output log range. These start addresses must also be stored in the pointer parameters of the FB1. The FB1 parameters listed below are used for further parameterization. There is no PROFIBUS version of the HHU. For this reason, an Ethernet connection is shown for the HHU in this diagram. The PROFIBUS slave address must be stored in the parameters MCP1BusAdr and MCP2BusAdr. Enter the pointer to the configured diagnostic address (e.g. P#A8190.0) in MCPxStatRec.

![Diagram of PROFIBUS connection on the MPI/DP port](image)

**Figure 13-16 PROFIBUS connection on the MPI/DP port**

<table>
<thead>
<tr>
<th>Relevant parameters (FB1)</th>
<th>MCP</th>
<th>HHU</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCPNum = 1 or 2 (number of MCPs)</td>
<td>MCP1In, MCP1Out</td>
<td>HHU = 5 (via CP 840D sl)</td>
</tr>
<tr>
<td>MCP1In</td>
<td>MCP2In</td>
<td>BHGIn</td>
</tr>
<tr>
<td>MCP1Out</td>
<td>MCP2Out</td>
<td>BHGOut</td>
</tr>
<tr>
<td>MCP1StatSend (n.r.)</td>
<td>MCP2StatSend (n.r.)</td>
<td>BHGStatSend</td>
</tr>
<tr>
<td>MCP1StatRec</td>
<td>MCP2StatRec</td>
<td>BHGStatRec</td>
</tr>
<tr>
<td>MCP1BusAdr</td>
<td>MCP2BusAdr</td>
<td>BHGInLen</td>
</tr>
<tr>
<td>MCP1Timeout</td>
<td>MCP2Timeout</td>
<td>BHGOutLen</td>
</tr>
<tr>
<td>MCP1Cycl (n.r.)</td>
<td>MCP2Cycl (n.r.)</td>
<td>BHGTimeout (n.r.)</td>
</tr>
<tr>
<td>MCPMPI = FALSE</td>
<td>MCP1BusType = b#16#44</td>
<td>BHGCycl (n.r.)</td>
</tr>
<tr>
<td>MCP1Stop</td>
<td>MCP2Stop</td>
<td>BHGRecGBZNo (n.r.)</td>
</tr>
<tr>
<td>MCPSDB210 = FALSE</td>
<td>MCPCopyDB77 = FALSE</td>
<td>BHGRecObjNo (n.r.)</td>
</tr>
<tr>
<td>BHGSendGDNo (n.r.)</td>
<td>BHGSendGBZNo (n.r.)</td>
<td>BHGSendObjNo (n.r.)</td>
</tr>
</tbody>
</table>
MCP failure normally switches the PLC to the STOP state. If this is undesirable, then OB82, OB86 can be used to avoid a PLC stop. The basic program has, as standard, the OB82 and OB86 call. FC5 is called in these OBs. This FC5 checks whether the failed slave is an MCP. If this is the case, no PLC stop is triggered. Setting MCPxStop:= TRUE causes the basic program to deactivate the MCP as a slave via SFC12. If the PLC does not switch to the stop state following the failure or fault of the MCP, an alarm message will be generated via the basic program. The alarm is deleted when the station returns.

### PROFINET connection (MCPBusType = 6)

In case of PROFINET connection of the MCP, this component must be parameterized in the hardware configuration setting of STEP 7. The MCP is coupled with the PROFINET module of the CPU.

When parameterizing the MCP in HW Config, the addresses should be placed in the input and output mapping area. These start addresses must also be stored in the pointer parameters (MCPxIn and MCPxOut) of FB1. This is because signals are transferred between the MCP and basic program via these parameters. The MCP is also monitored using parameter MCPxIn. This is the reason why parameter MCPxBusAdr is not relevant for this MCP version.

Enter the pointer to the configured diagnostic address (e.g. P#A8190.0) in MCPxStatRec.

The PROFINET MCP has its own type which should be applied for parameter MCPBusType.

The FB1 parameters listed below are used for further parameterization. There is no PROFIBUS version of the HHU. An Ethernet port for the HHU is shown in the diagram.

---

**Relevant parameters (FB1)**

<table>
<thead>
<tr>
<th>MCP</th>
<th>HHU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BHGMPI = FALSE</td>
</tr>
<tr>
<td></td>
<td>BHGStop</td>
</tr>
</tbody>
</table>

---

**Figure 13-17** PROFINET connection
### Relevant parameters (FB1)

<table>
<thead>
<tr>
<th>MCP</th>
<th>HHU</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCPNum = 1 or 2 (number of MCPs)</td>
<td>HHU = 5 (via CP 840D sl)</td>
</tr>
<tr>
<td>MCP1In</td>
<td>MCP2In</td>
</tr>
<tr>
<td>MCP1Out</td>
<td>MCP2Out</td>
</tr>
<tr>
<td>MCP1StatSend (n.r.)</td>
<td>MCP2StatSend (n.r.)</td>
</tr>
<tr>
<td>MCP1StatRec</td>
<td>MCP2StatRec</td>
</tr>
<tr>
<td>MCP1BusAdr (n.r.)</td>
<td>MCP2BusAdr (n.r.)</td>
</tr>
<tr>
<td>MCP1Timeout</td>
<td>MCP2Timeout</td>
</tr>
<tr>
<td>MCP1Cycl</td>
<td>MCP2Cycl</td>
</tr>
<tr>
<td>MCP1MPI = FALSE</td>
<td>BHGCycl (n.r.)</td>
</tr>
<tr>
<td>MCPStop</td>
<td>MCPStop</td>
</tr>
<tr>
<td>MCPBusType = b#16#36</td>
<td>BHGRecGBZNo (n.r.)</td>
</tr>
<tr>
<td>(as in the figure as example)</td>
<td>(6 = PROFINET for MCP1)</td>
</tr>
<tr>
<td></td>
<td>(3 = PROFIBUS for MCP2)</td>
</tr>
<tr>
<td>MCPSDB210 = FALSE</td>
<td>BHGSendGDNo (n.r.)</td>
</tr>
<tr>
<td>MCPCopyDB77 = FALSE</td>
<td>BHGSendGBZNo (n.r.)</td>
</tr>
<tr>
<td></td>
<td>BHGSendObjNo (n.r.)</td>
</tr>
<tr>
<td></td>
<td>BHGMPI = FALSE</td>
</tr>
<tr>
<td></td>
<td>BHGStop</td>
</tr>
</tbody>
</table>

MCP failure normally switches the PLC to the STOP state. If this is undesirable, then OB82, OB86 can be used to avoid a PLC stop. The basic program has, as standard, the OB82 and OB86 call. FC5 is called in these OBs. This FC5 checks whether the failed slave is an MCP. If this is the case, no PLC stop is triggered. The input address at parameter MCPxIn is of significance when monitoring for MCPxIn failure.

Setting MCPxStop:= TRUE causes the basic program to deactivate the MCP as a slave via SFC12. If the PLC does not switch to the stop state following the failure or fault of the MCP, an alarm message will be generated via the basic program. The alarm is deleted when the station returns.

#### 13.10.12 Switchover of machine control panel, handheld unit

Only Ethernet variants support switchover/deactivation of an operator component (MCP or HHU) as standard.

**PROFIBUS variant**

With PROFIBUS variants, this functionality is only possible to a limited extent and with additional user effort.

With the PROFIBUS variant of the MCP, the data area of the DB77 for specified MCP1, MCP2 or HHU can be used for the MCP pointer on FB1. The MCP slave bus address must be set correctly under MCPxBusAdr as this is used as the basis for monitoring. A user program copy
routine must copy the signals of the active MCP from the I/O area configured in HW Config to DB77. This enables a number of MCPs on the PROFIBUS to be switched via signals. Set the MCPxStop parameter to TRUE for the switchover phase from one MCP to another.

Control signals

Parameters MCP1Stop, MCP2Stop and BHGStop can be used to stop communication with individual components (parameter setting = 1). This stop or activation of communication can be applied in the current cycle. However, the change in value must be implemented through the symbolic notation of the parameters and not by means of another FB1 call.

Example: Stopping the transfer from the 1st machine control panel:

```
SET;
S gp_par.MCP1Stop;
```

Setting parameters MCP1Stop, MCP2Stop, BHGStop also results in a suppression or deletion of alarms 400260 to 400262.

Switchover of the bus address

If an existing communication connection to an operator component (MCP or HHU) is to be cancelled and a new communication connection established to a different component (MCP or HHU) with a different communication address, proceed as follows:

1. Stop the communication of the operator component to be disconnected: Parameter MCP1Stop, MCP2Stop or BHGStop = 1
2. The communication is stopped when the following applies: DB10, DBX104.0, .1 or .2 == 0
3. Change the bus address:
   - MCP: FB1 parameter MCP1BusAdr or MCP2BusAdr = <bus address of the new operator component>
   - HHU (Ethernet variant): FB1 parameter BHGRecGDNo = <bus address of the new operator component>
4. Enable the communication (possible in the same PLC cycle as point 3): Parameter MCP1Stop, MCP2Stop or BHGStop = 0
5. The communication with the new component is active when the following applies: : DB10, DBX104.0, .1 or .2 == 1

Switching off the LED flashing of an Ethernet MCP

After a Power On, MCPs generally indicate the completion of the power-up and waiting for a connection to be established by flashing LEDs. The flashing of the LEDs can be switched off as described in the following. Presently, this behavior cannot be retentively stored on the MCP.

Requirement

MCP firmware as of V02.02.04
Setting for switching off the flashing

The Send status must be set in MCPxStop before the start of communication with the MCP. Before the start of communication means either during power-up (OB100) or during cyclic operation (OB1) before the setting of DB7 parameter MCPxStop = FALSE

Setting the Send status: FB1 parameter MCPxStatSend, bit 30 = 0 and bit 31 = 1

There is no feedback of the current status.

Example

Extract from OB100: (based on the example for MCP1)

CALL "RUN_UP", "gp_par"
...
MCP1StatSend := P#E 8.0
...
//deactivate MCP flashing
SET
R A 11.6
S A 11.7
...

13.11 SPL for Safety Integrated

Rather than being a function of the basic program, SPL is a user function. The basic program makes a data block (DB18) available for Safety SPL signals and runs a data comparison to ensure the consistency of SPL program data in the NCK.

References:
/FBSI/ Description of Functions Safety Integrated

13.12 Assignment overview

13.12.1 Assignment: NCK/PLC interface

The values of the NC/PLC interface for SINUMERIK 840D sl are described in detail in:

References:
Lists sl (Buch2)
13.12.2 Assignment: FB/FC

<table>
<thead>
<tr>
<th>Number</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB15</td>
<td>Basic program</td>
</tr>
<tr>
<td>FB1, FC2, FC3, FC5</td>
<td>Basic program</td>
</tr>
<tr>
<td>FC0 ... 29</td>
<td>Reserved for Siemens</td>
</tr>
<tr>
<td>FB0 ... 29</td>
<td>Reserved for Siemens</td>
</tr>
<tr>
<td>FC30 ... 999</td>
<td>Free for user assignment</td>
</tr>
<tr>
<td>FB30 ... 999</td>
<td>Free for user assignment</td>
</tr>
<tr>
<td>FC1000 ... 1023</td>
<td>Reserved for Siemens</td>
</tr>
<tr>
<td>FB1000 ... 1023</td>
<td>Reserved for Siemens</td>
</tr>
<tr>
<td>FC1024 ... upper limit</td>
<td>Free for user assignment</td>
</tr>
<tr>
<td>FB1024 ... upper limit</td>
<td>Free for user assignment</td>
</tr>
</tbody>
</table>

1) The actual upper limit of the block number (FB/FC) depends on the PLC CPU on which the selected NCU is located.

Note
Values of FC, FB see "Memory requirements of the basic PLC program (Page 897)".

13.12.3 Assignment: DB

Note
Only as many data blocks as are required according to the NC machine data configuration are set up.

Overview of data blocks

<table>
<thead>
<tr>
<th>DB no.</th>
<th>Name</th>
<th>Name</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reserved for Siemens</td>
<td>PLC-MELD</td>
<td>BP</td>
</tr>
<tr>
<td>2 ... 5</td>
<td>PLC-MELD</td>
<td>PLC messages</td>
<td>BP</td>
</tr>
<tr>
<td>6 ... 8</td>
<td>Basic program</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>NC-COMPILE</td>
<td>Interface for NC compile cycles</td>
<td>BP</td>
</tr>
<tr>
<td>10</td>
<td>NC INTERFACE</td>
<td>Central NCK interface</td>
<td>BP</td>
</tr>
<tr>
<td>11</td>
<td>Mode group 1</td>
<td>Interface mode group</td>
<td>BP</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>Computer link and transport system interface</td>
<td>BP</td>
</tr>
<tr>
<td>13 ... 14</td>
<td>Reserved for basic program</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Basic program</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>PI Service definition</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>Version identifier</td>
<td></td>
</tr>
</tbody>
</table>
### Overview of data blocks

<table>
<thead>
<tr>
<th>DB no.</th>
<th>Name</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Reserved for basic program</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>HMI interface</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>PLC machine data</td>
<td></td>
</tr>
<tr>
<td>21 ... 30</td>
<td>CHANNEL 1 ... n</td>
<td>Interface NC channels</td>
</tr>
<tr>
<td>31 ... 61</td>
<td>AXIS 1 ... m</td>
<td>Interfaces for axes/spindles or free for user assignment</td>
</tr>
<tr>
<td>62 ... 70</td>
<td>Free for user assignment</td>
<td></td>
</tr>
<tr>
<td>71 ... 74</td>
<td>Tool management</td>
<td></td>
</tr>
<tr>
<td>75 ... 76</td>
<td>M group decoding</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>Data block for MCP signals</td>
<td></td>
</tr>
<tr>
<td>78 ... 80</td>
<td>Reserved for Siemens</td>
<td></td>
</tr>
<tr>
<td>81 ... 999</td>
<td>see below: ShopMill, ManualTurn</td>
<td></td>
</tr>
<tr>
<td>1000 ... 1099</td>
<td>Reserved for Siemens</td>
<td></td>
</tr>
<tr>
<td>1100 ... High limit</td>
<td>Free for user assignment</td>
<td></td>
</tr>
</tbody>
</table>

1) The actual upper limit of the block number (DB) depends on the PLC CPU on which the selected NCU is located. The data blocks of channels, axes/spindles and tool management functions that are not activated may be assigned as desired by the user.

### Note

The data blocks of channels, axes/spindles and tool management functions that are not activated may be assigned as required by the user.

### 13.12.4 Assignment: Timers

<table>
<thead>
<tr>
<th>Timer No.</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 0 ... T 512 1)</td>
<td>User area</td>
</tr>
</tbody>
</table>

1) The actual upper limit of the timer number (DB) depends on the PLC CPU on which the selected NCU is located.
13.13 PLC functions for HMI

13.13.1 Program selection from the PLC

Function
Preselected programs/workpieces can be selected for machining by the NC via the PLC/HMI interface.
The preselection is realized by entering programs/workpiece in files (these are known as PLC program lists (*.ppl)).

Preconditions
The following machine data must be set to allow the HMI to process tasks:
MD9106 $MM_SERVE_EXTCALL_PROGRAMS
In order to activate a sector-specific PLC program list, you must set the appropriate machine data and at least the protection level password:

- **Area User**
  - MD51041 $MN_ENABLE_PROGLIST_USER = 1
  - Protective level password: 3 (users)
  - Program list: /user/sinumerik/hmi/plc/programlist/plc_proglist_user.ppl

- **Area Manufacturer (OEM)**
  - MD51043 $MN_ENABLE_PROGLIST_MANUFACT = 1
  - Protective level password: 1 (manufacturer)
  - Program list: /oem/sinumerik/hmi/plc/programlist/plc_proglist_manufacturer.ppl

Structure of a program list
A program list is a text file. Each line contains the following information:
<program number> <program path><program name> [CH=<channel number>]

- **Program number**
The program numbers which may be used in a program list depend on the sector:
  - user: 1 - 100
  - Individual (oem_i): 101 - 200 (only SINUMERIK 828D)
  - Manufacturer (OEM): 201 - 255

- **Channel number**
Specifying the channel number "CH=<channel number>" is optional. It is only required if the NC has more than one channel.
• Program path
  The program path must be completely specified in absolute terms.
  For specifying the program path, see:
  References
  Programming Manual, Work Planning, Chapter "File and Program Administration" >
  "Program memory" > "Addressing the files of the program memory"

• Channel number
  The channel number (1, 2, 3, ...) only has to be specified if more than one channel is
  configured in the NC.

The following excerpt as example shows the structure of the user program list:

Program list: plc_proglist_user.ppl
1 //DEV2:/MPFDir/PROG_01.MPF CH=1
2 //DEV2:/MPFDir/PROG_01.MPF CH=2

Generating entries in a program list
The entries in a program list (*.ppl) can be directly edited in the file or entered in screen forms
in the user interface.

• Via the user interface for the user area
  Operating area "Program Manager" > "ETC key (">")" > "Prog. list"

• Via the user interface for the Manufacturer area
  Operating area "Commissioning" > "System data" > "ETC key (">")" > "Prog. list"

Program selection: Job interface

Note
The PLC may only request a new job if the last job has been acknowledged by the HMI:
DB19.DBB26 == 0

Program list
DB19.DBB16 = <number of the program list>

<table>
<thead>
<tr>
<th>Number</th>
<th>Program list</th>
</tr>
</thead>
<tbody>
<tr>
<td>129</td>
<td>/user/sinumerik/hmi/plc/programlist/plc_proglist_user.ppl</td>
</tr>
<tr>
<td>130</td>
<td>/oem_i/sinumerik/hmi/plc/programlist/plc_proglist_individual.ppl (only SINUMERIK 828D)</td>
</tr>
<tr>
<td>131</td>
<td>/oem/sinumerik/hmi/plc/programlist/plc_proglist_manufacturer.ppl</td>
</tr>
</tbody>
</table>

Program number
The program number refers to the programs contained in the selected program list.
DB19.DBB17 = <program number>

- user area: 1 - 100
- individual area: 101 - 200 (only SINUMERIK 828D)
- oem area: 201 - 255

**Requesting program selection**
DB19.DBX13.7 = 1

**Program selection: Acknowledgment interface**

**Job acknowledgment**
- DB19.DBX26.7 == 1 (selection identified)
- DB19.DBX26.3 == 1 (program is selected)
- DB19.DBX26.2 == 1 (error when selecting the program, see error ID DB19.DBB27)
- DB19.DBX26.1 == 1 ((job completed)

**Error detection**
DB19.DBB27 == <error ID>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>1</td>
<td>Invalid program list number (DB19.DBB16)</td>
</tr>
<tr>
<td>3</td>
<td>User specification Program list plc_proglst_main.ppl not found (only for DB19.DBB16 ≠ 1, 2, 3)</td>
</tr>
<tr>
<td>4</td>
<td>Invalid program number (DB19.DBB17)</td>
</tr>
<tr>
<td>5</td>
<td>Job list in the selected workpiece could not be opened.</td>
</tr>
<tr>
<td>6</td>
<td>Error in job list (job list interpreter returns error).</td>
</tr>
<tr>
<td>7</td>
<td>Job list interpreter returns empty job list.</td>
</tr>
</tbody>
</table>

**Program selection: Job processing**

A job to select a program is executed as follows:

1. Checking the acknowledgment byte: DB19.DBB26 == 0
   If the acknowledgment byte is not 0, then the last job has still not been completed.

2. Specifying the program list: DB19.DBB16

3. Specifying the program number: DB19.DBB17

4. Setting the request to select a program: DB19.DBX13.7 = 1

5. Evaluating the acknowledgment and error interface: DB19.DBB26 and .DBB27
   The order is still not completed on the HMI side as long as: DB19.DBX26.3 == 1 (active)
   The order has been completed on the HMI side if one of the two signals has been set:
   - DB19.DBX26.1 == 1 (OK)
   - DB19.DBX26.2 == 1 (error)
6. To complete the order, the program selection request must be reset: DB19.DBX13.7 = 0
7. The HMI signals that it is ready to accept a new order by resetting the acknowledgment byte: DB19.DBB26 == 0

See also
A2: Various NC/PLC interface signals and functions (Page 35)

13.13.2 Activating the key lock

The operator panel keyboard and a keyboard directly connected to the HMI can be locked using the following interface signal

- 1. HMI: DB19.DBX0.2 = <value>
- 2. HMI: DB19.DBX50.2 = <value>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Key lock inactive</td>
</tr>
<tr>
<td>1</td>
<td>Key lock active</td>
</tr>
</tbody>
</table>

13.13.3 Operating area numbers

The number of the active operating area is normally displayed in: DB19.DBB21

If the HMI monitor is active, the number of the active operating area is no longer displayed in DB19.DBB21, but instead in the user-specific configured area of the HMI monitor (Page 893).

<table>
<thead>
<tr>
<th>Operating area</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine</td>
<td>201</td>
</tr>
<tr>
<td>Parameters</td>
<td>205</td>
</tr>
<tr>
<td>Programming</td>
<td>203</td>
</tr>
<tr>
<td>Program Manager</td>
<td>202</td>
</tr>
<tr>
<td>Diagnostics</td>
<td>204</td>
</tr>
<tr>
<td>Commissioning</td>
<td>206</td>
</tr>
</tbody>
</table>

13.13.4 Screen numbers

The current screen number is normally displayed in: DB19.DBW24

If the HMI monitor is active, the current screen number is no longer displayed in DB19.DBW24, but instead in the user-specific configured area of the HMI monitor (Page 893).
Screen number ranges

The following screen number ranges are available:

- **JOG, manual machine** (Page 885)
- **Reference point approach** (Page 890)
- **MDA** (Page 890)
- **AUTOMATIC** (Page 890)
- **Parameters operating area** (Page 891)
- **Program operating area** (Page 892)
- **Program manager operating area** (Page 893)
- **Diagnostics operating area** (Page 893)

### 13.13.4.1 Screen numbers: JOG, manual machine

#### JOG mode

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Turning technology</strong></td>
<td></td>
</tr>
<tr>
<td>Cycle start screen for all screens that can be taken over</td>
<td>81</td>
</tr>
<tr>
<td><strong>Milling technology</strong></td>
<td></td>
</tr>
<tr>
<td>Cycle start screen for all screens that can be taken over</td>
<td>3</td>
</tr>
<tr>
<td><strong>Turning/milling</strong></td>
<td></td>
</tr>
<tr>
<td>Start screen</td>
<td>19</td>
</tr>
<tr>
<td>T,S,M</td>
<td>2</td>
</tr>
<tr>
<td>Set WO</td>
<td>21</td>
</tr>
<tr>
<td>Positioning</td>
<td>4</td>
</tr>
<tr>
<td>Face milling</td>
<td>18</td>
</tr>
<tr>
<td>Stock removal</td>
<td>80</td>
</tr>
<tr>
<td>Cycle start screen for all user screens</td>
<td>91</td>
</tr>
<tr>
<td>General settings</td>
<td>1</td>
</tr>
<tr>
<td>Multi-channel function settings</td>
<td>106</td>
</tr>
<tr>
<td>Collision avoidance settings</td>
<td>107</td>
</tr>
<tr>
<td>Measurement log settings</td>
<td>108</td>
</tr>
<tr>
<td>Swiveling</td>
<td>60</td>
</tr>
<tr>
<td>All G functions</td>
<td>100</td>
</tr>
<tr>
<td>Actual zoom value (MCS/WCS)</td>
<td>101</td>
</tr>
<tr>
<td>Thread synchronizing</td>
<td>102</td>
</tr>
<tr>
<td>Retract</td>
<td>103</td>
</tr>
<tr>
<td>Handwheel</td>
<td>104</td>
</tr>
<tr>
<td>Action synchronization</td>
<td>105</td>
</tr>
<tr>
<td><strong>Turning technology: Workpiece zero</strong></td>
<td></td>
</tr>
<tr>
<td>Workpiece zero (main menu)</td>
<td>30</td>
</tr>
<tr>
<td>Screen</td>
<td>Number</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>User screen</td>
<td>31</td>
</tr>
<tr>
<td>User screen</td>
<td>34</td>
</tr>
<tr>
<td>User screen</td>
<td>35</td>
</tr>
<tr>
<td>User screen</td>
<td>36</td>
</tr>
<tr>
<td>User screen</td>
<td>37</td>
</tr>
<tr>
<td>User screen</td>
<td>38</td>
</tr>
<tr>
<td>Measure edge Z</td>
<td>40</td>
</tr>
</tbody>
</table>

**Turning technology: Workpiece, measurement**

<table>
<thead>
<tr>
<th></th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measure tool (main menu)</td>
<td>50</td>
</tr>
<tr>
<td>Manual X or user screen</td>
<td>51</td>
</tr>
<tr>
<td>Manual Y</td>
<td>71</td>
</tr>
<tr>
<td>Manual Z or user screen</td>
<td>52</td>
</tr>
<tr>
<td>Zoom or user screen</td>
<td>53</td>
</tr>
<tr>
<td>User screen</td>
<td>54</td>
</tr>
<tr>
<td>User screen</td>
<td>55</td>
</tr>
<tr>
<td>Probe calibration X or user screen</td>
<td>56</td>
</tr>
<tr>
<td>Probe calibration Z or user screen</td>
<td>57</td>
</tr>
<tr>
<td>Automatic length in Z</td>
<td>58</td>
</tr>
<tr>
<td>Automatic length in Y</td>
<td>73</td>
</tr>
<tr>
<td>Automatic length in X</td>
<td>59</td>
</tr>
</tbody>
</table>

**Milling technology: Workpiece zero**

<table>
<thead>
<tr>
<th></th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workpiece zero (main menu)</td>
<td>30</td>
</tr>
<tr>
<td>Measure edge X</td>
<td>5</td>
</tr>
<tr>
<td>Measure edge X</td>
<td>22</td>
</tr>
<tr>
<td>Measure edge Z</td>
<td>23</td>
</tr>
<tr>
<td>User screen</td>
<td>7</td>
</tr>
<tr>
<td>Align edge or user screen</td>
<td>31</td>
</tr>
<tr>
<td>Distance 2 edges or user screen</td>
<td>32</td>
</tr>
<tr>
<td>Right-angled corner</td>
<td>33</td>
</tr>
<tr>
<td>Any corner or user screen</td>
<td>8</td>
</tr>
<tr>
<td>Rectangular pocket</td>
<td>34</td>
</tr>
<tr>
<td>1 hole or user screen</td>
<td>9</td>
</tr>
<tr>
<td>2 holes</td>
<td>35</td>
</tr>
<tr>
<td>3 holes</td>
<td>36</td>
</tr>
<tr>
<td>4 holes</td>
<td>37</td>
</tr>
<tr>
<td>Rectangular spigot</td>
<td>38</td>
</tr>
<tr>
<td>1 circular spigot or user screen</td>
<td>10</td>
</tr>
<tr>
<td>2 circular spigots</td>
<td>39</td>
</tr>
<tr>
<td>3 circular spigots</td>
<td>40</td>
</tr>
<tr>
<td>4 circular spigots</td>
<td>41</td>
</tr>
<tr>
<td>Set up level</td>
<td>42</td>
</tr>
<tr>
<td>Probe length calibration or user screen</td>
<td>11</td>
</tr>
</tbody>
</table>
## Probe radius calibration

**Number**

12

### Milling technology: Workpiece, measurement

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measure tool (main menu)</td>
<td>50</td>
</tr>
<tr>
<td>Measure length, manual (with milling tool) or measure length in X, manual (with turning tool) or user screen</td>
<td>16</td>
</tr>
<tr>
<td>Measure length in Y, manual (with turning tool)</td>
<td>74</td>
</tr>
<tr>
<td>Measure length in Z, manual (with turning tool)</td>
<td>24</td>
</tr>
<tr>
<td>Measure diameter, manual or user screen</td>
<td>17</td>
</tr>
<tr>
<td>Measure length, automatic (with milling tool) or measure length in X, automatic (with turning tool) or user screen</td>
<td>13</td>
</tr>
<tr>
<td>Measure length in Y, automatic (with turning tool)</td>
<td>75</td>
</tr>
<tr>
<td>Measure length in Z, automatic (with turning tool)</td>
<td>25</td>
</tr>
<tr>
<td>Measure diameter, automatic or user screen</td>
<td>14</td>
</tr>
<tr>
<td>User screen</td>
<td>51</td>
</tr>
<tr>
<td>Probe calibration or user screen</td>
<td>15</td>
</tr>
<tr>
<td>Fixed point calibration or user screen</td>
<td>52</td>
</tr>
</tbody>
</table>

### RunMyScreens (only for set JobShopIntegration)

- User screen behind horizontal 1st softkey                              | 96     |
- User screen behind horizontal 2nd softkey                              | 98     |
- User screen behind horizontal 3rd softkey                              | 99     |
- User screen behind horizontal 4th softkey                              | 94     |
- User screen behind horizontal 5th softkey                              | 95     |
- User screen behind horizontal 6th softkey                              | 92     |
- User screen behind horizontal 7th softkey                              | 97     |
- User screen behind horizontal 8th softkey                              | 90     |
- User screen behind horizontal 9th softkey                              | 83     |
- User screen behind horizontal 10th softkey                             | 82     |
- User screen behind horizontal 11th softkey                             | 93     |
- User screen behind horizontal 12th softkey                             | 84     |
- User screen behind horizontal 13th softkey                             | 85     |
- User screen behind horizontal 14th softkey                             | 86     |
- User screen behind horizontal 15th softkey                             | 87     |
- User screen behind horizontal 16th softkey                             | 88     |

### JOG mode, manual machine

**DB19.DBB24**

<table>
<thead>
<tr>
<th>Screen</th>
<th>Screen number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turning/milling</td>
<td></td>
</tr>
<tr>
<td>Taper turning</td>
<td>61</td>
</tr>
<tr>
<td>Angle milling</td>
<td>62</td>
</tr>
<tr>
<td>Screen</td>
<td>Screen number</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Stop</td>
<td>63</td>
</tr>
<tr>
<td>Straight line</td>
<td>1300</td>
</tr>
<tr>
<td>Straight all axes</td>
<td>1330</td>
</tr>
<tr>
<td>Straight X alpha</td>
<td>1340</td>
</tr>
<tr>
<td>Straight Z alpha</td>
<td>1350</td>
</tr>
<tr>
<td>Circle</td>
<td>1360</td>
</tr>
<tr>
<td>Drilling</td>
<td>1400</td>
</tr>
<tr>
<td>Center drilling</td>
<td>1410</td>
</tr>
<tr>
<td>Drilling, thread centered</td>
<td>1420</td>
</tr>
<tr>
<td>Drilling, centering</td>
<td>1433</td>
</tr>
<tr>
<td>Drilling, drilling</td>
<td>1434</td>
</tr>
<tr>
<td>Drilling, reaming</td>
<td>1435</td>
</tr>
<tr>
<td>Drilling, boring</td>
<td>1436</td>
</tr>
<tr>
<td>Drilling, deep hole drilling</td>
<td>1440</td>
</tr>
<tr>
<td>Drilling, deep hole drilling 2</td>
<td>1441</td>
</tr>
<tr>
<td>Drilling, tapping</td>
<td>1453</td>
</tr>
<tr>
<td>Drill thread milling</td>
<td>1455</td>
</tr>
<tr>
<td>Positions</td>
<td>1473</td>
</tr>
<tr>
<td>Position row</td>
<td>1474</td>
</tr>
<tr>
<td>Position grid</td>
<td>1477</td>
</tr>
<tr>
<td>Position frame</td>
<td>1478</td>
</tr>
<tr>
<td>Position circle</td>
<td>1475</td>
</tr>
<tr>
<td>Position pitch circle</td>
<td>1479</td>
</tr>
<tr>
<td>Obstacle</td>
<td>1476</td>
</tr>
<tr>
<td>Turning</td>
<td>1500</td>
</tr>
<tr>
<td>Turning, stock removal 1</td>
<td>1513</td>
</tr>
<tr>
<td>Turning, stock removal 2</td>
<td>1514</td>
</tr>
<tr>
<td>Turning, stock removal 3</td>
<td>1515</td>
</tr>
<tr>
<td>Turning, groove 1</td>
<td>1523</td>
</tr>
<tr>
<td>Turning, groove 2</td>
<td>1524</td>
</tr>
<tr>
<td>Turning, groove 3</td>
<td>1525</td>
</tr>
<tr>
<td>Turning, undercut form E</td>
<td>1533</td>
</tr>
<tr>
<td>Turning, undercut form F</td>
<td>1534</td>
</tr>
<tr>
<td>Turning, undercut thread DIN</td>
<td>1535</td>
</tr>
<tr>
<td>Turning, undercut thread</td>
<td>1536</td>
</tr>
<tr>
<td>Turning, thread, longitudinal</td>
<td>1543</td>
</tr>
<tr>
<td>Turning, thread, taper</td>
<td>1544</td>
</tr>
<tr>
<td>Turning, thread, facing</td>
<td>1545</td>
</tr>
<tr>
<td>Turning, thread, chain</td>
<td>1546</td>
</tr>
<tr>
<td>Turning, cut-off</td>
<td>1550</td>
</tr>
<tr>
<td>Milling</td>
<td>1600</td>
</tr>
<tr>
<td>Milling, face milling</td>
<td>1610</td>
</tr>
<tr>
<td>Screen</td>
<td>Screen number</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Milling, rectangular pocket</td>
<td>1613</td>
</tr>
<tr>
<td>Milling, circular pocket</td>
<td>1614</td>
</tr>
<tr>
<td>Milling, rectangular spigot</td>
<td>1623</td>
</tr>
<tr>
<td>Milling, circular spigot</td>
<td>1624</td>
</tr>
<tr>
<td>Milling, longitudinal groove</td>
<td>1633</td>
</tr>
<tr>
<td>Milling, circumferential groove</td>
<td>1634</td>
</tr>
<tr>
<td>Milling, open groove</td>
<td>1635</td>
</tr>
<tr>
<td>Milling, multi-edge</td>
<td>1640</td>
</tr>
<tr>
<td>Milling, thread milling</td>
<td>1454</td>
</tr>
<tr>
<td>Milling, engraving</td>
<td>1670</td>
</tr>
<tr>
<td>Contour turning</td>
<td>1200</td>
</tr>
<tr>
<td>Contour turning, new contour / last contour</td>
<td>1210</td>
</tr>
<tr>
<td>Contour turning, stock removal along contour</td>
<td>1220</td>
</tr>
<tr>
<td>Contour turning, contour grooving</td>
<td>1230</td>
</tr>
<tr>
<td>Contour turning, contour plunge turning</td>
<td>1240</td>
</tr>
<tr>
<td>Contour milling</td>
<td>1100</td>
</tr>
<tr>
<td>Contour milling, new contour / last contour</td>
<td>1110</td>
</tr>
<tr>
<td>Contour milling, path milling</td>
<td>1120</td>
</tr>
<tr>
<td>Contour milling, centering</td>
<td>1130</td>
</tr>
<tr>
<td>Contour milling, rough drilling</td>
<td>1140</td>
</tr>
<tr>
<td>Contour milling, contour pocket</td>
<td>1150</td>
</tr>
<tr>
<td><strong>Turning technology: Simulation</strong></td>
<td></td>
</tr>
<tr>
<td>Side view</td>
<td>1740</td>
</tr>
<tr>
<td>Front view</td>
<td>1750</td>
</tr>
<tr>
<td>3D view</td>
<td>1760</td>
</tr>
<tr>
<td>2-window view</td>
<td>1770</td>
</tr>
<tr>
<td>Half section</td>
<td>1780</td>
</tr>
<tr>
<td><strong>Turning technology: Simultaneous recording</strong></td>
<td></td>
</tr>
<tr>
<td>Side view</td>
<td>1741</td>
</tr>
<tr>
<td>Front view</td>
<td>1751</td>
</tr>
<tr>
<td>3D view</td>
<td>1761</td>
</tr>
<tr>
<td>2-window view</td>
<td>1771</td>
</tr>
<tr>
<td>Machine space</td>
<td>1791</td>
</tr>
<tr>
<td>Half section</td>
<td>1781</td>
</tr>
<tr>
<td><strong>Milling technology: Simulation</strong></td>
<td></td>
</tr>
<tr>
<td>Top view</td>
<td>1742</td>
</tr>
<tr>
<td>3D view</td>
<td>1760</td>
</tr>
<tr>
<td>From the front</td>
<td>1744</td>
</tr>
<tr>
<td>From the rear</td>
<td>1746</td>
</tr>
<tr>
<td>From the Left</td>
<td>1748</td>
</tr>
<tr>
<td>From the right</td>
<td>1752</td>
</tr>
<tr>
<td>Half section</td>
<td>1780</td>
</tr>
</tbody>
</table>
13.13.4.2 Screen numbers: Reference point approach

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual zoom value MCS/WCS</td>
<td>101</td>
</tr>
</tbody>
</table>

13.13.4.3 Screen numbers: MDA

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDA</td>
<td>20</td>
</tr>
<tr>
<td>All G functions</td>
<td>100</td>
</tr>
<tr>
<td>Actual zoom value MCS/WCS</td>
<td>101</td>
</tr>
<tr>
<td>Handwheel</td>
<td>104</td>
</tr>
<tr>
<td>Action synchronization</td>
<td>105</td>
</tr>
<tr>
<td>Program control</td>
<td>210</td>
</tr>
<tr>
<td>Settings</td>
<td>250</td>
</tr>
</tbody>
</table>

13.13.4.4 Screen numbers: AUTOMATIC

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>200</td>
</tr>
<tr>
<td>Overstore</td>
<td>202</td>
</tr>
<tr>
<td>Program control</td>
<td>210</td>
</tr>
<tr>
<td>Block search</td>
<td>220</td>
</tr>
<tr>
<td>General settings</td>
<td>250</td>
</tr>
<tr>
<td>Multi-channel function settings</td>
<td>106</td>
</tr>
<tr>
<td>Collision avoidance settings</td>
<td>107</td>
</tr>
<tr>
<td>All G functions</td>
<td>100</td>
</tr>
</tbody>
</table>
### Screen

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual zoom value MCS/WCS</td>
<td>101</td>
</tr>
<tr>
<td>Handwheel</td>
<td>104</td>
</tr>
<tr>
<td>Action synchronization</td>
<td>105</td>
</tr>
</tbody>
</table>

**Turning technology: Simultaneous recording**

<table>
<thead>
<tr>
<th>View</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Side view</td>
<td>243</td>
</tr>
<tr>
<td>Front view</td>
<td>244</td>
</tr>
<tr>
<td>3D view</td>
<td>245</td>
</tr>
<tr>
<td>2-window view</td>
<td>246</td>
</tr>
<tr>
<td>Machine space</td>
<td>247</td>
</tr>
<tr>
<td>Half section</td>
<td>253</td>
</tr>
</tbody>
</table>

**Milling technology: Simultaneous recording**

<table>
<thead>
<tr>
<th>View</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top view</td>
<td>242</td>
</tr>
<tr>
<td>3D view</td>
<td>244</td>
</tr>
<tr>
<td>From the front</td>
<td>248</td>
</tr>
<tr>
<td>From the rear</td>
<td>249</td>
</tr>
<tr>
<td>From the Left</td>
<td>251</td>
</tr>
<tr>
<td>From the right</td>
<td>252</td>
</tr>
<tr>
<td>Machine space</td>
<td>247</td>
</tr>
<tr>
<td>Half section</td>
<td>253</td>
</tr>
<tr>
<td>Turning view</td>
<td>254</td>
</tr>
</tbody>
</table>

### 13.13.4.5 Screen numbers: Parameters operating area

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool list</td>
<td>600</td>
</tr>
<tr>
<td>Tool wear</td>
<td>610</td>
</tr>
<tr>
<td>User tool list</td>
<td>620</td>
</tr>
<tr>
<td>Magazine</td>
<td>630</td>
</tr>
</tbody>
</table>

**Work offset**

<table>
<thead>
<tr>
<th>Work offset</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work offset, active</td>
<td>642</td>
</tr>
<tr>
<td>Work offset, overview</td>
<td>643</td>
</tr>
<tr>
<td>Work offset, basic</td>
<td>644</td>
</tr>
<tr>
<td>Work offset, G54 - G509</td>
<td>645</td>
</tr>
<tr>
<td>Details of work offset, active, overview, basic or G54 - G509</td>
<td>647</td>
</tr>
</tbody>
</table>

**User variable**

<table>
<thead>
<tr>
<th>User variable</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>R parameters</td>
<td>650</td>
</tr>
<tr>
<td>Global GUD 1 (SGUD)</td>
<td>660</td>
</tr>
<tr>
<td>Global GUD 2 (MGUD)</td>
<td>661</td>
</tr>
<tr>
<td>Global GUD 3 (UGUD)</td>
<td>662</td>
</tr>
<tr>
<td>Global GUD 4</td>
<td>663</td>
</tr>
<tr>
<td>Global GUD 5</td>
<td>664</td>
</tr>
<tr>
<td>Global GUD 6</td>
<td>665</td>
</tr>
</tbody>
</table>
### Screen Numbers

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global GUD 7</td>
<td>666</td>
</tr>
<tr>
<td>Global GUD 8</td>
<td>667</td>
</tr>
<tr>
<td>Global GUD 9</td>
<td>668</td>
</tr>
<tr>
<td>Channel GUD 1 (SGUD)</td>
<td>690</td>
</tr>
<tr>
<td>Channel GUD 2 (MGUD)</td>
<td>691</td>
</tr>
<tr>
<td>Channel GUD 3 (UGUD)</td>
<td>692</td>
</tr>
<tr>
<td>Channel GUD 4</td>
<td>693</td>
</tr>
<tr>
<td>Channel GUD 5</td>
<td>694</td>
</tr>
<tr>
<td>Channel GUD 6</td>
<td>695</td>
</tr>
<tr>
<td>Channel GUD 7</td>
<td>696</td>
</tr>
<tr>
<td>Channel GUD 8</td>
<td>697</td>
</tr>
<tr>
<td>Channel GUD 9</td>
<td>698</td>
</tr>
<tr>
<td>Local LUD</td>
<td>681</td>
</tr>
<tr>
<td>Local LUD/PUD</td>
<td>684</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Setting data</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Working area limitation</td>
<td>671</td>
</tr>
<tr>
<td>Spindle data</td>
<td>670</td>
</tr>
<tr>
<td>Spindle chuck data</td>
<td>672</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ctrl-Energy</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl-Energy, main menu</td>
<td>6170</td>
</tr>
<tr>
<td>Ctrl-Energy, analysis</td>
<td>6171</td>
</tr>
<tr>
<td>Ctrl-Energy, profiles</td>
<td>6172</td>
</tr>
<tr>
<td>Ctrl-Energy, analysis graphic</td>
<td>6176</td>
</tr>
<tr>
<td>Ctrl-Energy, analysis long-term measurement</td>
<td>6177</td>
</tr>
<tr>
<td>Ctrl-Energy, analysis details</td>
<td>6179</td>
</tr>
<tr>
<td>Ctrl-Energy, compare measurements</td>
<td>6178</td>
</tr>
</tbody>
</table>

### 13.13.4.6 Screen numbers: Program operating area

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Turning technology: Simulation</strong></td>
<td></td>
</tr>
<tr>
<td>Side view</td>
<td>413</td>
</tr>
<tr>
<td>Front view</td>
<td>414</td>
</tr>
<tr>
<td>3D view</td>
<td>415</td>
</tr>
<tr>
<td>2-window view</td>
<td>416</td>
</tr>
<tr>
<td>Half section</td>
<td>423</td>
</tr>
</tbody>
</table>

| **Milling technology: Simulation** | |
| Top view                         | 412    |
| 3D view                          | 414    |
| From the front                   | 418    |
| From the rear                    | 419    |
| From the Left                    | 421    |
### 13.13.4.7 Screen numbers: Program manager operating area

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>From the right</td>
<td>422</td>
</tr>
<tr>
<td>Half section</td>
<td>423</td>
</tr>
<tr>
<td>Turning view</td>
<td>424</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC directory</td>
<td>300</td>
</tr>
<tr>
<td>Local drive</td>
<td>325</td>
</tr>
<tr>
<td>USB / configured drive1</td>
<td>330</td>
</tr>
<tr>
<td>Configured drive2</td>
<td>340</td>
</tr>
<tr>
<td>Configured drive3</td>
<td>350</td>
</tr>
<tr>
<td>Configured drive4</td>
<td>360</td>
</tr>
<tr>
<td>Configured drive5</td>
<td>383</td>
</tr>
<tr>
<td>Configured drive6</td>
<td>384</td>
</tr>
<tr>
<td>Configured drive7</td>
<td>385</td>
</tr>
<tr>
<td>Configured drive8</td>
<td>386</td>
</tr>
</tbody>
</table>

### 13.13.4.8 Screen numbers: Diagnostics operating area

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm list</td>
<td>500</td>
</tr>
<tr>
<td>Messages</td>
<td>501</td>
</tr>
<tr>
<td>Alarm log</td>
<td>502</td>
</tr>
<tr>
<td>NC/PLC variable</td>
<td>503</td>
</tr>
</tbody>
</table>

### 13.13.5 HMI monitor

#### Function

The HMI monitor is an 8-byte data area in a freely selectable data block, in which the HMI can provide the following data for the PLC user program:

- Operating area numbers (Page 884)
- Screen numbers (Page 884)

#### Parameterization

The data area is configured using the following display machine data:

MD9032 $MM_HMI_MONITOR = "string"
with "string" = "DB<DB number>.DBB<byte address>"

Note

Even byte address

The data area must start at an even byte address.

Structure of the data area

<table>
<thead>
<tr>
<th>Byte</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>EB n + 0</td>
<td>Active SINUMERIK operating area</td>
</tr>
<tr>
<td>EB n + 1</td>
<td>Reserved</td>
</tr>
<tr>
<td>EB n + 2</td>
<td>Current screen number</td>
</tr>
<tr>
<td>EB n + 3</td>
<td></td>
</tr>
<tr>
<td>EB n + 4</td>
<td>Reserved</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>EB n + 7</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Supplementary conditions

When the HMI monitoring is active, the following PLC/HMI interface signals are no longer processed:

- DB19.DBB10 (PLC hardkeys)
- DB19.DBB21 (active SINUMERIK operating area)
- DB19.DBW24 (current screen number)

Example

Assumptions

- Current operating area: "Machine", number: 201
- Actual screen: "AUTOMATIC" start screen, number: 200
- PLC data area: DB60.DBB10

Parameterization

- MD9032 $MM_HMI_MONITOR = "DB60.DBB10"

Values in the data area

- DB60.DBB10: 201
- DB60.DBW12: 200
13.14 PLC functions for drive components on the integrated PROFIBUS

13.14.1 Overview

Using the function described below, input and output data from drive components on the integrated PROFIBUS can be consistently, cyclically read and written from the PLC user program of the hardware PLC. The following boundary conditions must be observed:

- For reading / writing input/output data, the system functions SFC14 / SFC15 must be used.
- The reading / writing of input/output data is always over the entire slot length.
- The function only supports cyclical, non-equidistant data transfer.
- An output slot may not already be occupied on the NC side; (e.g. output slots of drives).

13.14.2 Performing a start-up

Preconditions

Before the function is put into operation on the NC side, the following preconditions must be satisfied:

- The drive components on the integrated PROFIBUS of the NCU must be fully configured using SIMATIC STEP 7, HW Config.
- The configuration is loaded into the PLC.

NC machine data

The start addresses of the slots to be transferred cyclically are to be entered into the following machine data:

- Input slots:
  \[ \text{MD10520} \, \text{MN_PLCINTERN_LOGIC_ADDRESS_IN}[<\text{Index}>] = <\text{Slot address}> \]
- Output slots:
  \[ \text{MD10525} \, \text{MN_PLCINTERN_LOGIC_ADDRESS_OUT}[<\text{Index}>] = <\text{Slot address}> \]

<Index>: 0, 1, 2, ... (Max no. of slots) – 1
<Slot address>: The slot address parameterized in HW Config

Note

Maximum quantity of data

The sum of all data to be cyclically transferred in inward and outward directions may not currently exceed 2,048 bytes.
13.14.3 Example

Determining slot addresses

After selecting the DP Slave "SINAMICS_Integrated" on the integrated PROFIBUS "PROFIBUS Integrated: DP master system (3)" in the station window of HW Config, its PROFIdrive message frame and associated slot addresses are displayed in the detailed view.

- Message frame 136: Drive
- Message frame 391: Control Unit
- Message frame 370: Infeed

The infeed slot addresses required for parameterization in the NC machines are:

- Input slot: Slot 31, address 6514
- Output slot: Slot 32, address 6514

![Figure 13-18 Infeed PROFIBUS message frames]

Setting NC machine data

- Input slots: 
  MD10520 $MN_PLCINTERN_LOGIC_ADDRESS_IN[ 0 ] = 6514
- Output slots: 
  MD10525 $MN_PLCINTERN_LOGIC_ADDRESS_OUT[ 0 ] = 6514

Controlling the ALM using FB390 "ALM_Control"

The SINUMERIK hardware PLC is connected to the CU320 of SINAMICS S120 through the internal PROFIBUS. The ALM is connected to the CU via DRIVE-CLiQ.

In SINAMICS S120, a control and status message frame can be defined for each module (the CU, ALM, motor modules, etc.). If this is performed for an ALM, this can be switched on and off from the PLC user program.
The SIMATIC S7 block FB390 "ALM_Control" checks the status of the ALM and enables the user to switch it on or off.

A description of the block and an example project are available for download under the following link to Industry Online Support:

http://support.automation.siemens.com/WW/view/de/49515414

13.15 Memory requirements of the basic PLC program

The basic program consists of basic and optional functions. The **basic functions** include cyclic signal exchange between the NC and PLC. The **Options** include e.g. the FCs, which can be used, if needed.

The table below lists the memory requirements for the basic functions and the options. The data quoted represent guide values, the actual values depend on the current software version.

| Memory requirements of blocks for SINUMERIK 840D sl |
|-------------------------------|-----------------|-------------------|
| **Block Type, No.** | **Function** | **Remark** | **Block size (bytes)** |
| **Basic functions in basic program** | | | |
| FB1, FB15 | must be loaded / on CF card | | 52182 |
| FC2, 3, 5, 12 | Must be loaded | | 470 |
| DB4, 5, 7, 8 | Must be loaded | | 1006 |
| DB2, 3, 17 | Are generated by the BP | | 632 |
| OB1, 40, 100, 82, 86 | Must be loaded | | 398 |
| **Total** | | | 55698 |
| **PLC/NCK, PLC/HMI interface** | | | |
| DB10 | PLC/NCK signals | Must be loaded | 262 |
| DB11 | Signals PLC/Mode group | Is generated by BP | 56 |
| DB19 | PLC/HMI signals | Is generated by BP | 434 |
| DB21 to 30 | PLC/channel signals | Are generated by BP as a function of NCMD: for each DB | 416 |
| DB31 to 61 | PLC / axis or spindle signals | Are generated by BP as a function of NCMD: for each DB | 148 |
| **Basic program options** | | | |
| Machine control panel | | | |
| FC19 | Transfer of MCP signals, M variant | Must be loaded when M variant of MCP is installed | 92 |
### Basic program options

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC25</td>
<td>Transfer of MCP signals, T variant</td>
<td>Must be loaded when T variant of MCP is installed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>92</td>
</tr>
<tr>
<td>FC24</td>
<td>Transfer of MCP signals, slim variant</td>
<td>Must be loaded when slim variant of MCP is installed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>FC26</td>
<td>Transfer of MCP signals, HT8 variant</td>
<td>Must be loaded for HT8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>68</td>
</tr>
</tbody>
</table>

**Handheld unit**

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC13</td>
<td>Display control HHU</td>
<td>Can be loaded for handheld units</td>
</tr>
<tr>
<td></td>
<td></td>
<td>144</td>
</tr>
</tbody>
</table>

**Error/operating messages**

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC10</td>
<td>Acquisition FM/BM</td>
<td>Load when FM / BM is used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>66</td>
</tr>
</tbody>
</table>

**ASUB**

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC9</td>
<td>ASUB start</td>
<td>Load when PLC ASUBs are used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128</td>
</tr>
</tbody>
</table>

### Basic program options

**Star/delta changeover**

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC17</td>
<td>Star/delta switchover of MSD</td>
<td>Load for star/delta switchover</td>
</tr>
<tr>
<td></td>
<td></td>
<td>114</td>
</tr>
</tbody>
</table>

**Spindle control**

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC18</td>
<td>Spindle control</td>
<td>Load for spindle control from PLC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>132</td>
</tr>
</tbody>
</table>

**PLC/NC communication**

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB2</td>
<td>Read NC variable</td>
<td>Load for Read NC variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>76</td>
</tr>
<tr>
<td>DBn</td>
<td>Read NC variable</td>
<td>One instance DB per FB2 call:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>270</td>
</tr>
<tr>
<td>FB3</td>
<td>Write NC variable</td>
<td>Load for Write NC variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>76</td>
</tr>
<tr>
<td>DBm</td>
<td>Write NC variable</td>
<td>One instance DB per FB3 call:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>270</td>
</tr>
<tr>
<td>FB4</td>
<td>PI services</td>
<td>Load for PI services</td>
</tr>
<tr>
<td></td>
<td></td>
<td>76</td>
</tr>
<tr>
<td>DBo</td>
<td>PI services</td>
<td>One instance DB per FB4 call:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>130</td>
</tr>
<tr>
<td>DB16</td>
<td>PI services description</td>
<td>Load for PI services</td>
</tr>
<tr>
<td></td>
<td></td>
<td>618</td>
</tr>
<tr>
<td>FB5</td>
<td>Read GUD variables</td>
<td>Load for PI services</td>
</tr>
<tr>
<td></td>
<td></td>
<td>76</td>
</tr>
<tr>
<td>DBp</td>
<td>Read GUD variables</td>
<td>One instance DB per FB5 call:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>166</td>
</tr>
<tr>
<td>DB15</td>
<td>General communication</td>
<td>Global data block for communication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>146</td>
</tr>
<tr>
<td>FB7</td>
<td>PI services 2</td>
<td>Load for PI services</td>
</tr>
<tr>
<td></td>
<td></td>
<td>76</td>
</tr>
<tr>
<td>DBo</td>
<td>PI services 2</td>
<td>One instance DB per FB4 call:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>144</td>
</tr>
<tr>
<td>FC21</td>
<td>Transfer</td>
<td>Load with dual-port RAM, ...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>164</td>
</tr>
</tbody>
</table>

**M to N**

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB9</td>
<td>Switchover M to N</td>
<td>Load with M to N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58</td>
</tr>
</tbody>
</table>

**Safety Integrated**

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB10</td>
<td>Safety relay</td>
<td>Load with Safety option</td>
</tr>
<tr>
<td></td>
<td></td>
<td>74</td>
</tr>
<tr>
<td>FB11</td>
<td>Brake test</td>
<td>Load with Safety option</td>
</tr>
<tr>
<td></td>
<td></td>
<td>76</td>
</tr>
<tr>
<td>DB18</td>
<td>Safety data</td>
<td>DB for Safety</td>
</tr>
<tr>
<td></td>
<td></td>
<td>308</td>
</tr>
</tbody>
</table>
### Basic program options

<table>
<thead>
<tr>
<th>Code</th>
<th>Function</th>
<th>Remark</th>
<th>Block size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC7</td>
<td>Transfer function turret</td>
<td>Load for tool management option</td>
<td>84</td>
</tr>
<tr>
<td>FC8</td>
<td>Transfer function</td>
<td>Load for tool management option</td>
<td>132</td>
</tr>
<tr>
<td>FC22</td>
<td>Direction selection</td>
<td>Load, when direction selection is needed</td>
<td>138</td>
</tr>
<tr>
<td>DB71</td>
<td>Loading locations</td>
<td>Generated by BP as a function of NC MD</td>
<td>40 + 30*B</td>
</tr>
<tr>
<td>DB72</td>
<td>Spindles</td>
<td>Generated by BP as a function of NC MD</td>
<td>40 + 48*Sp</td>
</tr>
<tr>
<td>DB73</td>
<td>Revolver</td>
<td>Generated by BP as a function of NC MD</td>
<td>40 + 44*R</td>
</tr>
<tr>
<td>DB74</td>
<td>Basic function</td>
<td>Generated by BP as a function of NC MD</td>
<td>100 + (B + Sp + R) * 22</td>
</tr>
</tbody>
</table>

### Compile cycles

<table>
<thead>
<tr>
<th>Code</th>
<th>Function</th>
<th>Remark</th>
<th>Block size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB9</td>
<td>Interface PLC compile cycles</td>
<td>Is generated by BP as a function of NC option</td>
<td>436</td>
</tr>
</tbody>
</table>

### Example:

Based on the memory requirements in the table above, the memory requirements have been determined for two sample configurations (see table below).

<table>
<thead>
<tr>
<th>Block Type, No.</th>
<th>Function</th>
<th>Remark</th>
<th>Block size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Working memory</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Minimum configuration (1 spindle, 2 axes and T MCP)

<table>
<thead>
<tr>
<th>Block Type, No.</th>
<th>Function</th>
<th>Remark</th>
<th>Block size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>see above</td>
<td>Basic program, base</td>
<td></td>
<td>54688</td>
</tr>
<tr>
<td></td>
<td>Interface DBs</td>
<td></td>
<td>1612</td>
</tr>
<tr>
<td></td>
<td>MCP</td>
<td></td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>56392</td>
</tr>
</tbody>
</table>

#### Maximum configuration (2 channels, 4 spindles, 4 axes, T MCP)

<table>
<thead>
<tr>
<th>Block Type, No.</th>
<th>Function</th>
<th>Remark</th>
<th>Block size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>see above</td>
<td>Basic program, base</td>
<td></td>
<td>54688</td>
</tr>
<tr>
<td>see above</td>
<td>Interface DBs</td>
<td></td>
<td>2768</td>
</tr>
<tr>
<td>see above</td>
<td>MCP</td>
<td></td>
<td>92</td>
</tr>
<tr>
<td>see above</td>
<td>Error/operating messages</td>
<td></td>
<td>66</td>
</tr>
<tr>
<td>see above</td>
<td>ASUBs</td>
<td>1 ASUB initiation</td>
<td>128</td>
</tr>
<tr>
<td>see above</td>
<td>Concurrent axis</td>
<td>For 2 turrets</td>
<td>132</td>
</tr>
<tr>
<td>see above</td>
<td>PLC/NC communication</td>
<td>1 x read variable and 1 x write variable</td>
<td>838</td>
</tr>
<tr>
<td>see above</td>
<td>Tool management</td>
<td>2 turrets with one loading point each</td>
<td>674</td>
</tr>
</tbody>
</table>
### Maximum configuration (2 channels, 4 spindles, 4 axes, T MCP)

<table>
<thead>
<tr>
<th>see above</th>
<th>Compile cycles</th>
<th>436</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td></td>
<td>59822</td>
</tr>
</tbody>
</table>

#### 13.16 Basic conditions and NC VAR selector

#### 13.16.1 Supplementary conditions

#### 13.16.1.1 Programming and parameterizing tools

**Hardware**

For the PLCs used in SINUMERIK 840D sl, the following equipment is required for the programming devices or PCs:

<table>
<thead>
<tr>
<th></th>
<th>Minimum</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Pentium</td>
<td>Pentium</td>
</tr>
<tr>
<td>RAM (MB)</td>
<td>256</td>
<td>512 or more</td>
</tr>
<tr>
<td>Hard disk, free capacity (MB)</td>
<td>500</td>
<td>&gt; 500</td>
</tr>
<tr>
<td>Interfaces</td>
<td>MPI, Ethernet incl. cable, Memory card</td>
<td></td>
</tr>
<tr>
<td>Graphic</td>
<td>SVGA (1024*768)</td>
<td></td>
</tr>
<tr>
<td>Mouse</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Operating system</td>
<td>Windows 2000 /XP Professional, STEP 7 version 5.3 SP2 or higher</td>
<td></td>
</tr>
</tbody>
</table>

The required version of **STEP 7** can be installed on equipment meeting the above requirements in cases where the package has not already been supplied with the programming device.
The following functions are possible with this package:

- **Programming**
  - Editors and compilers for STL (complete scope of the language incl. SFB/SFC calls), LAD, FBD
  - Creation and editing of assignment lists (symbol editor)
  - Data block editor
  - Input and output of blocks ON/OFF line
  - Insertion of modifications and additions ON and OFF line
  - Transfer of blocks from programming device to the PLC and vice versa

- **Parameterizing**
  - Parameterizing tool **HW Config** for CPU and I/O device parameterization
  - **NetPro** parameterizing tool for setting the CPU communication parameters
  - Output of system data such as hardware and software version, memory capacity, I/O expansion/assignment

- **Testing and diagnostics (ONLINE)**
  - Variable status/forcing (I/Os, flags, data block contents, etc.)
  - Status of individual blocks
  - Display of system states (ISTACK, BSTACK, system status list)
  - Display of system messages
  - Trigger PLC stop / restart / general reset from the PG
  - Compress PLC

- **Documentation**
  - Printout of individual or all blocks
  - Allocation of symbolic names (also for variables in data blocks)
  - Input and output of comments within each block
  - Printout of test and diagnostics displays
  - Hardcopy function
  - Cross-reference list
  - Program overview
  - Assignment plan I/O/M/T/Z/D
• Archiving of utility routines
  – Allocation of the output states of individual blocks
  – Comparison of blocks
  – Rewiring
  – STEP 5 → STEP 7 converter

• Option packages
  – Programming in S7-HIGRAPH, S7-GRAPH, SCL.
    These packages can be ordered from the SIMATIC sales department.
  – Additional packages for configuration modules (e.g. CP3425 → NCM package)

**Note**
More information about possible functions can be found in SIMATIC catalogs and STEP 7 documentation.

### 13.16.1.2 SIMATIC documentation required

**References:**
- System description SIMATIC S7
- S7-300 instruction list
- Programming with STEP 7
- User Manual STEP 7
- Programming manual STEP 7; designing of user programs
- Reference manual STEP 7; Instructions list AWL
- Reference manual STEP 7; Ladder Diagram KOP
- Reference manual STEP 7; Default and system functions
- Manual STEP 7; Conversion of STEP 5 programs
- STEP 7 overall index
- Manual CPU 317-2DP

### 13.16.1.3 Relevant SINUMERIK documents

**References:**
- Commissioning Manual IBN CNC: NCK, PLC, Drive
- Operator Components and Networking Manual
- Function Manual Basic Functions
- Function Manual, Extended Functions:
- Function Manual, Special Functions
13.16.2 NC VAR selector

13.16.2.1 Overview

General

The PC application "NC VAR selector" retrieves the addresses of required NC variables and processes them for access in the PLC program (FB2/FB3). This enables the programmer to select NC variables from the entire range of NC variables, to store this selection of variables, to edit them by means of a code generator for the STEP 7 compiler and finally to store them as an ASCII file (*.AWL) in the machine CPU program. This process is shown in the figure "NC VAR selector".

For storing the files created by NC-VAR-selector a catalog is to be implemented via the Windows Explorer with any catalog name. The selected data of the NC-VAR selector (data.VAR and data.AWL files) must be stored in this catalog. Thereafter, the STL file is to be transferred and compiled via the menu option "Code" → "in STEP 7 Project". The "data.AWL" (STL data) file must then be inserted into the STEP 7 machine project via "Insert", "External Source" in the STEP 7 Manager. The source container must be selected in the manager for this purpose. This action stores this file in the project structure. Once the file has been transferred, these AWL (STL) files must be compiled with STEP 7.

Note

The latest NC VAR selector can be used for each NC software version (even earlier versions). For older NC software versions the variables can also be selected from the latest complete list. The data content in DB120 (default DB for variables) does not depend on the software status. That is, variables selected in an older software version need not be reselected when the software is upgraded.
After the "NC VAR selector" application has been started, select a list of variables of an NC variant (hard disk → file Ncv.mdb) to display all the variables contained in this list in a window.

The following ncv*.mdb variable list is available:

<table>
<thead>
<tr>
<th>Variables</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC variables including machine and setting data:</td>
<td>ncv_NcData.mdb</td>
</tr>
<tr>
<td>Parameters of the drive:</td>
<td>ncv_SinamicsServo.mdb</td>
</tr>
</tbody>
</table>

The user can also transfer variables to a second list (separate window). This latter selection of variables can then be stored in an ASCII file or edited as a STEP 7 source file (.awl) and stored.

After generating a PLC data block by means of the STEP 7 compiler, the programmer is able to read or write NCK variables via the basic program function blocks “PUT” and “GET” using the STEP 7 file.

The list of selected variables is also stored as an ASCII file (file extension .var).

The variable list supplied with the "NC VAR selector" tool is adapted to the current NC software version. This list does not contain any variables (GUD variables) defined by the user. These variables are processed by the function block FB5 in the basic program.

**Note**

The latest version of the "NC VAR selector" is capable of processing all previous NC software versions. It is, therefore, not necessary to install different versions of the "NC VAR selector" in parallel.
System features, supplementary conditions

The PC application "NC VAR selector" requires Windows 2000 or a higher operating system.

The assignment of names to variables is described in:
References:
/List sl (Book1); Chapter: Variables, or in the variables help file (integrated in NC VAR selector).

13.16.2.2 Description of functions

Overview

The figure below illustrates how the NC VAR selector is used within the STEP 7 environment.

![Diagram of NC VAR selector in STEP 7 environment]

The NC VAR selector is used to generate a list of selected variables from a list of variables and then to generate an *.awl file that can be compiled by the STEP 7 compiler.

- A *.awl file contains the names and alias names of the NC variables, as well as information about their address parameters. Any data block generated from this file will only contain the address parameters (10 bytes per parameter).
- The generated data blocks must always be stored in the machine-specific file storage according to STEP 7 specifications.
- To ensure that the parameterization of the GET/PUT (FB2/3) blocks with respect to NC addresses can be implemented with symbols, the freely assignable, symbolic name of the generated data block must be included in the STEP 7 symbol table.
Basic display / basic menu

After the NC VAR selector has been selected (started), the basic display with all input options (upper menu bar) appears on the screen. All other displayed windows are placed within the general window.

![Basic display with basic menu](image)

Project menu item

All operator actions associated with the project file (file of selected variables) are performed under this menu item.

Terminating the application

The application can be terminated by selecting the "End" option under the "Project" menu item.

Creating a new project

A new project (new file for selected variables) can be set up under the "Project" menu item. A window is displayed for the selected variables when "NEW" is selected. The file selection for the NC variable list is then displayed after a prompt (applies only if the NC variable list is not already open).
The selected variables are displayed in a window.

Open an already existing project

Select "Open" under the "Project" menu item to open an existing project (variables already selected). A file selection window is displayed allowing the appropriate project with extension ".var" to be selected.

If, after selection of the project, new variables are to be added, a complete list of NC variables must be selected. No complete list need be called if the user only wishes to delete variables from the project.

Storing a project

The variable list is stored using the "Project" > "Save" or "Save As..." menu items.
"Save" stores the variable list under a path, which is already specified. If the project path is not known, then the procedure is as for "Save As....".

"Save As..." displays a window in which the path for the project to be stored can be specified.

**Printing a project**

The "Print" command under the "Project" menu item can be selected to print a project file. The number of lines per page is selected under the "Print Setting" menu item. The default setting is 77 lines.

**Edit menu item**

The following operator actions are examples of those, which can be carried out directly with this menu item:

- Transfer variables
- Delete variables
- change alias names
- Find variables

These actions can also be canceled again under Edit.

**Undoing actions**

Operator actions relating to the creation of the project file (transfer variables, delete variables, change alias names) can be undone in this menu.

**NC variables menu item**

The basic list of all variables is saved in NC Var Selector path Data\Swxy (xy stands for software version no., e.g. SW 5.3:=xy=53). This list can be selected as an NC variables list. In case of SINUMERIK 840D sl the basic lists are present in the path Data\Swxy_sl.
Selecting an NC variable list

A list of all the NC variables for an NC version can now be selected and displayed via the "NC Variable List", "Select" menu item.

![Figure 13-24 Window with selected Complete List](image)

The field variables (e.g. axis area, T area data, etc.) are indicated by means of brackets ([ ]). Additional information must be specified here. When the variables are transferred to the project list, the additional information required is requested.

Displaying subsets

Double-click on any table field (with the exception of variable fields) to display a window in which filter criteria can be preset.

![Figure 13-25 Window with filter criteria for displaying list of variables](image)
There are three options:

- Display all data
- Input area, block and name (incl. combinations)
- Display MD/SE data number

The following wildcards can also be used:

* To extend the search criterion as required

**Example search criteria**

Name search criterion: CHAN* Found:

- CHAN_NAME
  - chanAlarm
  - chanStatus
  - channelName
  - chanAssignment

- Select variable
  A variable is selected by means of a simple mouse click and transferred to the window of selected variables by double-clicking. This action can also be undone under the "Edit" menu item.

**Alias name**

The variable names provided can be up to 32 characters in length. To make variables clearly identifiable in the data block to be generated, several ASCII characters are added to the selected name. However, the STEP 7 compiler recognizes only 24 ASCII characters as an unambiguous STEP 7 variable. Since it cannot be precluded that variable names can only be differentiated by the last 8 character positions, **ALIAS** names are used for names, which are too long. When a variable is selected, the length of the STEP 7 name to be used is, therefore, checked. If the name is longer than 24 characters, the user must enter an additional name, which is then used as the alias.

**In this case, the user must ensure that the alias name is unambiguous.**

Alias input can always be activated by the user in the "Options" menu. An alias name can then be entered every time a variable is transferred.

It is also possible to edit alias names at a later point in time by double-clicking on the S7 variable name field. This action can also be undone under the "Edit" menu item.
Scrolling

A scroll bar is displayed if it is not possible to display all variables in the window. The remaining variables can be reached by scrolling (page up/down).

Variables in multi-dimensional structures

If variables are selected from multi-dimensional structures, then the column and/or line number as well as the area number must be entered so that the variables can be addressed. The required numbers can be found in the NC variables documentation.

References:
Lists sl (Book1); Variables

By entering a zero (0) as the block number or the line or column index, it is possible to use the variable in the S7 PLC as a pointer to these data. When reading or writing these data via the functions "PUT" and "GET", the optional parameters "UnitX", "ColumnX" and "LineX" must be filled with the necessary information.
Delete variables

Variables are deleted in the window of selected variables by selecting the appropriate variables (single mouse click) and pressing the "Delete" key. No deletion action is taken with the double-click function. It is possible to select several variables for deletion (see Section "Example of search criteria > Selecting variables").

This action can also be undone under the "Edit" menu item.

Note

Deleting of variables results in a change of the absolute addresses of the pointer structures to the variables. When changing the variable selection, it is, therefore, absolutely necessary to generate one or several text files of all user blocks prior to the change. This is the only way to ensure that the assignment of the variables in FB "GET" or FB "PUT" remains correct, even after recompilation.

Storing a selected list

Once variables have been selected, they can be stored under a project name. The files are stored on a project-specific basis.

A window is displayed for the file to be stored. The project path and name for the file must be selected in the window.

![Figure 13-28 Window for project path and name of file to be stored](image)
Code generation

This menu item contains three selection options:
1. Settings (input of data block number to be generated) and other settings
2. Generate (create data block)
3. In the STEP 7 project (transferring the data block to a STEP 7 project)

Settings

Under this menu item, the DB number and the symbol for this DB number for which the code is created is entered.

Under the "Mass System" tab, a selection is made to determine how the unit system variables are calculated in the PLC.

Under the "Generate" tab, the project creation is defined for the relevant target system.

Generate

Under this menu item, the STEP 7 file from the selected variable list with extension ".awl" is set.

A file is generated when "Select" is clicked:

An .awl file that can be used as an input for the STEP 7 compiler.

A window opens, in which path and name for the .awl file to be generated must be specified for the file to be saved.

In STEP 7 project

The generated STL file is transferred to a selectable SIMATIC project (program path) and compiled. Furthermore, the symbol can also be transferred. This function is available as of STEP 7 Version 5.1. This process takes a longer time owing to the call of STEP 7. Before transferring a new STL file the file window of the STL file is to be closed in the LAD/FBD/STL editor.

Option menu item

The following can be selected under the "Option" menu item:

- The current language
- The mode for alias input (always / > 24 characters)

Help menu item

The information below can be viewed by selecting the corresponding submenu item:

- The Operating Manual
- The Description of Variables

The copyright and the version number can also be displayed.
13.16.2.3 Startup, installation

The Windows application "NC Var selector" is installed using the SETUP program supplied with the package.

13.17 Block descriptions

13.17.1 FB1: RUN_UP - basic program, start section

Function

The synchronization of NC and PLC is performed during startup. The data blocks for the NC/PLC user interface are created based on the NC configuration defined via machine data and the basic program parameters are verified for plausibility. In the event of an error, FB1 passes an error identifier to the diagnostic buffer and switches the PLC to the stop state.

"Restart" start-up mode

The integrated PLC only supports the start-up mode "Restart". After the basic system initialization, the organization block OB100 "Restart" is always run through first, followed by OB1 "Cyclic mode".

Input parameters

For parameterizing the basic program, only the respective relevant parameters of the function block FB1 must be written with user-specific values. The preset values in the instance data block DB7 of the FB1 do not need to be assigned. The function block FB1 must only be called in the organization block OB100.

Output parameters

The output parameters of function block FB1 can also be read from the cyclical part of the basic program. Two options are available for this purpose:

1. Direct access to the instance data block DB7 of FB1 in symbolic form.
   Example: "L gp_par.MaxChan", with "gp_par" as the symbolic name of DB7

2. A bit memory is assigned to an output parameter when FB1 is parameterized. The bit memory is then read in the basic program in order to determine the value of the output parameter.
   Example: "MaxChan":= MW 20

Note

For assigning the FB1 parameters for MCP and BHG, see "Configuration machine control panel, handheld unit, direct keys (Page 867)".
FUNCTION_BLOCK FB1
VAR_INPUT
MCPNum: INT:=1; // 0: No MCP
// 1: 1 MCP (default)
// 2: 2 MCPs
MCP1In: POINTER; // Start addr. MCP1 input signals
MCP1Out: POINTER; // Start addr. MCP1 output signals
MCP1StatSend: POINTER; // Status DW for sending MCP1
MCP1StatRec: POINTER; // Status DW for receiving MCP1
MCP1BusAdr: INT:=6; // default
MCP1Timeout: S5TIME:= S5T#700MS;
MCP1Cycl: S5TIME:= S5T#200MS;
MCP1Stop: BOOL:= FALSE;
MCP1NotSend: BOOL:= FALSE;
MCP2BusAdr: INT;
MCP2In: POINTER; // Start addr. MCP2 input signals
MCP2Out: POINTER; // Start addr. MCP2 output signals
MCP2StatSend: POINTER; // Status DW for sending MCP2
MCP2StatRec: POINTER; // Status DW for receiving MCP2
MCP2Timeout: S5TIME:= S5T#700MS;
MCP2Cycl: S5TIME:= S5T#200MS;
MCPMPI: BOOL:= FALSE;
MCP1Stop: BOOL:= FALSE;
MCP2Stop: BOOL:= FALSE;
MCP1NotSend: BOOL:= FALSE;
MCP2NotSend: BOOL:= FALSE;
MCPSDB210: BOOL:= FALSE;
MCPCopyDB77: BOOL:= FALSE;
MCPBusType: BYTE=B#16#0;
HHU: INT:=0; // Handheld unit interface
// 0: No HHU
// 1: HHU on MPI
// 2: HHU on OPI
BHGIN: POINTER; // Transmit data of the handheld unit
BHGOOut: POINTER; // Receive data of the handheld unit
BHGSStatSend: POINTER; // Status DW for sending HHU
BHGSStatRec: POINTER; // Status DW for receiving HHU
BHGINLen: BYTE:= B#16#6; // Input 6 bytes
BHGOOutLen: BYTE:= B#16#14; // Output 20 bytes
BHGTTimeout: S5TIME:= S5T#700MS;
BHGCycl: S5TIME:= S5T#100MS;
BHGRRecGDNo: INT:=2;
BHGRRecGBZNo: INT:=2;
BHGRRecObjNo: INT:=1;
BHGSendGDNo: INT:=2;
BHGSendGBZNo: INT:=1;
BHGSendObjNo: INT:=1;
BHGMPI: BOOL:= FALSE;
BHGSStop: BOOL:= FALSE;
BHGNotSend: BOOL:= FALSE;
NCCyclTimeout: S5TIME:= S5T#200MS;
NCRunupTimeout: S5TIME:= S5T#50S;
ListMDecGrp: INT:=0;
NCKomm: BOOL:= FALSE;
MMCToIF: BOOL:=TRUE;
HWheelMMC: BOOL:=TRUE; // Handwheel selection via HMI
ExtendAlMsg : BOOLEAN;
MsgUser: INT:=10; // Number of user areas in DB 2
UserIR: BOOL:= FALSE; // User programs in OB40,
// Observe local data expansion!
IRAuxfuT: BOOL:= FALSE; // Evaluate T function in OB40
IRAuxfuH: BOOL:= FALSE; // Evaluate H function in OB40
IRAuxfuE: BOOL:= FALSE; // Evaluate DL function in OB40
UserVersion: POINTER; // Pointer to string variable,
which is displayed in the version
display of the user interface

OpKeyNum : INT;
Op1KeyIn  POINTER;
Op1KeyOut : POINTER;
Op1KeyBusAdr : INT;
Op2KeyIn  POINTER;
Op2KeyOut : POINTER;
Op2KeyBusAdr : INT;
Op1KeyStop : BOOL;
Op2KeyStop : BOOL;
Op1KeyNotSend : BOOL;
Op2KeyNotSend : BOOL;
OpKeyBusType : BYTE;
IdentMcpBusAdr : INT;
IdentMcpProfilNo : BYTE;
IdentMcpBusType : BYTE;
IdentMcpStrobe : BOOL;

END_VAR

VAR_OUTPUT
MaxBAG: INT;
MaxChan: INT;
MaxAxis: INT;
ActivChan: ARRAY[1..10] OF BOOL;
### Description of formal parameters of SINUMERIK 840D sl

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCPNum:</td>
<td>INT</td>
<td>0, 1, 2</td>
<td>Number of active MCPs 0: No MCPs available</td>
</tr>
<tr>
<td>MCP1In:</td>
<td>POINTER</td>
<td>E0.0 ... E120.0 or M0.0 ... M248.0</td>
<td>Start address for input signals of relevant machine control panel</td>
</tr>
<tr>
<td>MCP2In:</td>
<td>POINTER</td>
<td>A0.0 ... A120.0 or M0.0 ... M248.0</td>
<td>Start address for output signals of relevant machine control panel</td>
</tr>
<tr>
<td>MCP1StatSend:</td>
<td>POINTER</td>
<td>A0.0 ... A124.0 or M0.0 ... M252.0</td>
<td>Only for Ethernet MCP: Switch off flashing (see Section &quot;Switch-over of machine control panel, handheld unit (Page 876)&quot;)</td>
</tr>
<tr>
<td>MCP2StatSend:</td>
<td>POINTER</td>
<td>A0.0 ... A124.0 or M0.0 ... M252.0</td>
<td>Only for Ethernet MCP: Switch off flashing (see Section &quot;Switch-over of machine control panel, handheld unit (Page 876)&quot;)</td>
</tr>
<tr>
<td>MCP1StatRec:</td>
<td>POINTER</td>
<td>A0.0 ... A124.0 or M0.0 ... M252.0</td>
<td>Currently no significance</td>
</tr>
<tr>
<td>MCP1BusAdr:</td>
<td>INT</td>
<td>1, 2, 3 ... 126</td>
<td>DP slave: PROFIBUS address</td>
</tr>
<tr>
<td>MCP2BusAdr:</td>
<td>INT</td>
<td>192, 193, 194 .. 223</td>
<td>Ethernet MCP: DIP setting</td>
</tr>
<tr>
<td>MCP1Timeout:</td>
<td>S5time</td>
<td>Recommendation: 700 ms</td>
<td>Cyclic sign-of-life monitoring for machine control panel</td>
</tr>
<tr>
<td>MCP2Timeout:</td>
<td>S5time</td>
<td>Recommendation: 200 ms</td>
<td>Relevant only for PROFIBUS</td>
</tr>
<tr>
<td>MCP1Cycl:</td>
<td>S5time</td>
<td>Recommendation: 200 ms</td>
<td>Relevant only for PROFIBUS</td>
</tr>
<tr>
<td>MCPMPI:</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Available owing to compatibility</td>
</tr>
<tr>
<td>MCP1Stop:</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>DP slave: Slave deactivated</td>
</tr>
<tr>
<td>MCP2Stop:</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>DP slave: Slave deactivated</td>
</tr>
<tr>
<td>Signal</td>
<td>Type</td>
<td>Value range</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------</td>
<td>--------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MCP1NotSend: MCP2NotSend:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE) 0: Send and receive operation activated 1: Receive machine control panel signals only</td>
</tr>
<tr>
<td>MCPSDB210:</td>
<td>I</td>
<td>BOOL</td>
<td>false Available owing to compatibility</td>
</tr>
<tr>
<td>MCPPCopyDB77:</td>
<td>I</td>
<td>BOOL</td>
<td>false Available owing to compatibility</td>
</tr>
<tr>
<td>MCPBusType:</td>
<td>I</td>
<td>BYTE</td>
<td>3, 4, 5, 6 b#16#yx: *Bus type MCP1: lower nibble (x) 3: PROFIBUS 4: PROFIBUS on the MPI/DP port 5: Ethernet 6: PROFINET Mixed mode is possible (see Chapter &quot;Configuration machine control panel, handheld unit, direct keys (Page 867)&quot;)</td>
</tr>
<tr>
<td>HHU:</td>
<td>I</td>
<td>INT</td>
<td>0, 5 Handheld unit interface 0: No HHU 5: HHU on Ethernet</td>
</tr>
<tr>
<td>BHGIn:</td>
<td>I</td>
<td>POINTER</td>
<td>E0.0 ... E124.0 or M0.0 ... M252.0 or DBn DBX0.0 ... DBXm.0 Start address PLC receive data from HHU</td>
</tr>
<tr>
<td>BHGOut:</td>
<td>I</td>
<td>POINTER</td>
<td>A0.0 ... A124.0 or M0.0 ... M252.0 or DBn DBX0.0 ... DBXm.0 Start address PLC transmit data to HHU</td>
</tr>
<tr>
<td>BHGStatSend:</td>
<td>I</td>
<td>POINTER</td>
<td>A0.0 ... A124.0 or M0.0 ... M252.0 or DBn DBX0.0 ... DBXm.0 Available owing to compatibility</td>
</tr>
<tr>
<td>BHGStatRec:</td>
<td>I</td>
<td>POINTER</td>
<td>A0.0 ... A124.0 or M0.0 ... M252.0 or DBn DBX0.0 ... DBXm.0 Available owing to compatibility</td>
</tr>
<tr>
<td>BHGInLen:</td>
<td>I</td>
<td>BYTE</td>
<td>HHU default: B#16#6 (6 Byte) Available owing to compatibility</td>
</tr>
<tr>
<td>BHGOutLen:</td>
<td>I</td>
<td>BYTE</td>
<td>HHU default: B#16#14 (20 Byte) Available owing to compatibility</td>
</tr>
<tr>
<td>BHGTimeout:</td>
<td>I</td>
<td>S5time</td>
<td>Recommendation: 700 ms Available owing to compatibility</td>
</tr>
<tr>
<td>BGHCycl:</td>
<td>I</td>
<td>S5time</td>
<td>Recommendation: 100 ms Available owing to compatibility</td>
</tr>
<tr>
<td>BHGRecGDNo:</td>
<td>I</td>
<td>INT</td>
<td>HHU default: 2 Ethernet DIP switch Available owing to compatibility</td>
</tr>
<tr>
<td>BHGRecGBZNo:</td>
<td>I</td>
<td>INT</td>
<td>HHU default: 2 Available owing to compatibility</td>
</tr>
<tr>
<td>BHGRecObjNo:</td>
<td>I</td>
<td>INT</td>
<td>HHU default: 1 Available owing to compatibility</td>
</tr>
</tbody>
</table>
### Signal Table

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BHGSendGDNo:</td>
<td>I</td>
<td>INT</td>
<td>HHU default: 2 Available owing to compatibility</td>
</tr>
<tr>
<td>BHGSendGBZNo:</td>
<td>I</td>
<td>int</td>
<td>HHU default: 1 Available owing to compatibility</td>
</tr>
<tr>
<td>BHGSendObjNo:</td>
<td>I</td>
<td>INT</td>
<td>HHU default: 1 Available owing to compatibility</td>
</tr>
<tr>
<td>BHGMPI:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE) Available owing to compatibility</td>
</tr>
<tr>
<td>BHGStop:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE) 0: Start transmission of handheld unit signals 1: Stop transmission of handheld unit signals</td>
</tr>
<tr>
<td>BHGNotSend:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE) 0: Send and receive operation activated 1: Receive handheld unit signals only</td>
</tr>
<tr>
<td>NCCycITimeout:</td>
<td>I</td>
<td>S5time</td>
<td>Recommendation: 200 ms Cyclic sign-of-life monitoring NCK</td>
</tr>
<tr>
<td>NCRunupTimeout:</td>
<td>I</td>
<td>S5time</td>
<td>Recommendation: 50 s Power-up monitoring NCK</td>
</tr>
<tr>
<td>ListMDecGrp:</td>
<td>I</td>
<td>INT</td>
<td>0, 1, 2 ... 16 Activation of expanded M group decoding 0: Not active 1...16: Number of M groups</td>
</tr>
<tr>
<td>NCKomm:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE) PLC NC communication services FB2, 3, 4, 5, 7 1: Active</td>
</tr>
<tr>
<td>MMCToIF:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE) Transmission of HMI signals to interface (modes, program control etc.) 1: Active</td>
</tr>
<tr>
<td>HWheelMMC:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE) 0: Handwheel selection via HMI 1: Handwheel selection via user program</td>
</tr>
<tr>
<td>ExtendAIMsg:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE) Activation of extensions error and operating messages of the FC10 (see section “Extensions of the PLC alarms via block FC10&quot; in Chapter &quot;Interface PLC/HMI (Page 842)”)</td>
</tr>
<tr>
<td>MsgUser:</td>
<td>I</td>
<td>INT</td>
<td>0, 1, 2 ... 64 Number of user areas (DB2) – depends on parameter &quot;ExtendAIMsg&quot;</td>
</tr>
<tr>
<td>UserIR:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE) Local data expansion OB40 required for processing of signals from the user</td>
</tr>
<tr>
<td>IRAuxfuT:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE) Evaluate T function in OB40</td>
</tr>
<tr>
<td>IRAuxfuH:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE) Evaluate H function in OB40</td>
</tr>
<tr>
<td>IRAuxfuE:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE) Evaluate DL function in OB40</td>
</tr>
<tr>
<td>UserVersion:</td>
<td>I</td>
<td>POINTER</td>
<td>P#DBn.DBXx.0 Pointer to string variable, which is displayed in the version display of the user interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The following written format applies to the string variable (max. 41 characters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;&lt;&quot;&lt;Name&gt; &lt;Version xx.xx.xx&gt; &lt;Date yy/mm/dd&gt;&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: &quot;Test-Version 07.06.02 13/06/04&quot;</td>
</tr>
<tr>
<td>Signal</td>
<td>Type</td>
<td>Value range</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OpKeyNum :</td>
<td>I</td>
<td>INT</td>
<td>0, 1, 2</td>
</tr>
<tr>
<td>Op1KeyIn: Op2KeyIn:</td>
<td>I</td>
<td>POINTER</td>
<td>P#Ex.0 or P#Mx.0 or P#Dbn.DByx.0.</td>
</tr>
<tr>
<td>Op1KeyOut : Op2KeyOut:</td>
<td>I</td>
<td>POINTER</td>
<td>P#Ax.0 or P#Mx.0 or P#Dbn.DByx.0.</td>
</tr>
<tr>
<td>Op1KeyBusAdr: Op2KeyBusAdr:</td>
<td>I</td>
<td>INT</td>
<td>1, 2, 3 ... 191</td>
</tr>
<tr>
<td>Op1KeyStop: Op2KeyStop:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
</tr>
<tr>
<td>Op1KeyNotSend: Op2KeyNotSend:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
</tr>
<tr>
<td>OpKeyBusType :</td>
<td>I</td>
<td>BYTE</td>
<td>b#16#55</td>
</tr>
<tr>
<td>IdentMcpBusAdr :</td>
<td>I</td>
<td>INT</td>
<td>1, 2, 3 ... 254</td>
</tr>
<tr>
<td>IdentMcpProfilNo :</td>
<td>I</td>
<td>BYTE</td>
<td>0, 1</td>
</tr>
<tr>
<td>IdentMcpBusType :</td>
<td>I</td>
<td>BYTE</td>
<td>b#16#5</td>
</tr>
<tr>
<td>MaxBAG:</td>
<td>O</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
</tr>
<tr>
<td>MaxChan:</td>
<td>O</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
</tr>
<tr>
<td>MaxAxis:</td>
<td>O</td>
<td>INT</td>
<td>1, 2, 3 ... 31</td>
</tr>
<tr>
<td>ActivChan:</td>
<td>O</td>
<td>ARRAY[1...10] OF BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
</tr>
<tr>
<td>ActivAxis:</td>
<td>O</td>
<td>ARRAY [1..31] OF BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
</tr>
<tr>
<td>UDigest :</td>
<td>O</td>
<td>INT</td>
<td>0,</td>
</tr>
<tr>
<td>UHex:</td>
<td>O</td>
<td>INT</td>
<td>---</td>
</tr>
<tr>
<td>UReal :</td>
<td>O</td>
<td>INT</td>
<td>---</td>
</tr>
<tr>
<td>IdentMcpType :</td>
<td>O</td>
<td>BYTE</td>
<td>0, B#16#80, B#16#81, B#16#82 ... B#16#89</td>
</tr>
<tr>
<td>IdentMcpLengthIn :</td>
<td>O</td>
<td>BYTE</td>
<td>---</td>
</tr>
<tr>
<td>IdentMcpLengthOut :</td>
<td>O</td>
<td>BYTE</td>
<td>---</td>
</tr>
</tbody>
</table>
If an error occurs during communication with a machine control panel (MCP) or handheld unit (HHU), the following alarms are displayed on the HMI and the input signals (MCP1In, MCP2In or BHGIn) are set to the value zero:

**Error case: MCP / HHU (840D sl)**

- Alarm 400260: "MCP 1 failure"
- Alarm 400261: "MCP 2 failure"
- Alarm 400262: "HHU failure"

If resynchronization is possible between PLC and MCP/HHU, communication is resumed, the error message on the HMI is deleted by the basic program, and process values are transferred to the input signals (MCP1In, MCP2In or BHGIn) again.

**Example: FB1 call in OB100 (840D sl)**

```plaintext
ORGANIZATION_BLOCK OB100
VAR_TEMP
  OB100_EV_CLASS : BYTE ;
  OB100_STRTUP : BYTE ;
  OB100_PRIORITY : BYTE ;
  OB100_OB_NUMBR : BYTE ;
  OB100_RESERVED_1 : BYTE ;
  OB100_RESERVED_2 : BYTE ;
  OB100_STOP : WORD;
  OB100_RESERVED_3 : WORD;
  OB100_RESERVED_4 : WORD;
  OB100_DATE_TIME : DATE_AND_TIME;
END_VAR
BEGIN
  CALL FB1, DB7( // FB1 call, instance DB: DB7
    MCPNum := 1,
    MCP1In := P#E0.0,
    MCP1Out := P#A0.0,
    MCP1StatSend := P#A8.0,
    MCP1StatRec := P#A12.0,
    MCP1BusAdr := 6,
    MCP1Timeout := S5T#700MS,
    MCP1Cycl := S5T#200MS,
    NC-CyclTimeout := S5T#200MS,
    NC-RunupTimeout := S5T#50S);
  // INSERT USER PROGRAM HERE
END_ORGANIZATION_BLOCK
```

---

P3: Basic PLC program for SINUMERIK 840D sl

13.17 Block descriptions

Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2

921
13.17.2 FB2: GET - read NC variable

Function

The FB2 "GET" function block is used to read variables from the NC area.

In order to reference the NC variables, they are first selected with the "NC VAR selector" tool and generated as STL source in a data block. A name must then be assigned to this data block in the S7 symbol table. When calling FB2, the variable addresses are transferred in the following form: Parameter "Addr1" to "Addr8" = "<DB name>.<S7 name>"

Request for reading NC variables

Call of FB2 with positive edge change, parameter "Req" = 0 → 1

S7 names of the NC variables: Parameter "Addr1" to "Addr8" = "NCVAR".<S7 name>"

Pointer for writing the variable values: Parameters "RD1" to "RD8" = "P#<Address>"

Completion of the read request

Read request successfully completed: Parameter "Done" == 1.

Read request completed with error:"Parameter "Error" == 1, error cause in parameter "State"

Prerequisites

- Release of the NC/PLC communication by OB100, FB1 parameter "NCKomm" = 1
- For the data block DB120 (data interface), the S7 Symbol Editor must be used to assign a symbol (default: NCVAR) in the S7 symbol table of the S7 project. Via this symbol, the NC variable is then specified in the FB2 parameters "Addr<x>", e.g. "ADDR1: = "NCVAR".<NC variable>"

General conditions

- FB2 has multi-instance capability.
- Every call of FB2 must be assigned a separate instance DB from the user area.
- When channel-specific variables are read, only variables from exactly one channel may be addressed via "Addr1" to "Addr8" if FB2 is called.
- When drive-specific variables are read, only variables from exactly one SERVO drive object may be addressed via "Addr1" to "Addr8" if FB2 is called. The SERVO drive object must be assigned to a machine axis of the NC. The line index corresponds to the logical drive number.
- In a read job, only variables from the same area, channel, or drive object can be read.

Note

Error case

When reading variables from different channels or drive objects, or simultaneously from a channel and a drive object, an error message is output:

- "Error" == TRUE
- "State" == W#16#02
Variable addressing

For some NC variables, it is necessary to select "Area no." and/or "Line" or "Column" from the NC VAR selector. It is possible to select a basic type, i.e. "Area no.", "Line" and "Column" are preassigned "0". The values of the "Area no.", "Line" and "Column" specified by the NC VAR selector are checked for a "0" in FB2. If an NC-VAR selector value == "0", the corresponding value of the FB2 parameter is adopted. To do this, the FB2 parameters "Unit<x>", "Column<x>" and "Line<x>", with <x> = 1 - 8, must be written before FB2 is called.

<table>
<thead>
<tr>
<th>FB2 parameter</th>
<th>NC VAR selector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit</td>
<td>Area no.</td>
</tr>
<tr>
<td>Column</td>
<td>Column</td>
</tr>
<tr>
<td>Line</td>
<td>Line</td>
</tr>
</tbody>
</table>

Variables within one group can be combined in a job:

<table>
<thead>
<tr>
<th>Group</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C[1]</td>
</tr>
<tr>
<td>2</td>
<td>C[2]</td>
</tr>
<tr>
<td>3</td>
<td>V[.]</td>
</tr>
</tbody>
</table>

The same rules apply for channels 3 to 10 as for group 1 and group 2 shown in the example.

Note

The number of usable variables can be less than eight when simultaneously reading several variables of the "String" type.

Declaration of the function

```
FUNCTION_BLOCK FB2
VAR_INPUT
```
13.17 Block descriptions

```
Req :     BOOL;
NumVar :  INT;
Addr1 :   ANY;
Unit1 :   BYTE;
Column1 : WORD;
Line1 :   WORD;
Addr2 :   ANY;
Unit2 :   BYTE;
Column2 : WORD;
Line2 :   WORD;
Addr3 :   ANY;
Unit3 :   BYTE;
Column3 : WORD;
Line3 :   WORD;
Addr4 :   ANY;
Unit4 :   BYTE;
Column4 : WORD;
Line4 :   WORD;
Addr5 :   ANY;
Unit5 :   BYTE;
Column5 : WORD;
Line5 :   WORD;
Addr6 :   ANY;
Unit6 :   BYTE;
Column6 : WORD;
Line6 :   WORD;
Addr7 :   ANY;
Unit7 :   BYTE;
Column7 : WORD;
Line7 :   WORD;
Addr8 :   ANY;
Unit8 :   BYTE;
Column8 : WORD;
Line8 :   WORD;
END_VAR
VAR_OUTPUT
Error :    BOOL;
NDR :      BOOL;
State :    WORD;
END_VAR
```
VAR_IN_OUT
RD1 : ANY;
RD2 : ANY;
RD3 : ANY;
RD4 : ANY;
RD5 : ANY;
RD6 : ANY;
RD7 : ANY;
RD8 : ANY;
END_VAR

Description of formal parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Typ e</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req:</td>
<td>I</td>
<td>BOOL</td>
<td>---</td>
<td>Job start with positive signal edge</td>
</tr>
<tr>
<td>NumVar:</td>
<td>I</td>
<td>INT</td>
<td>1 ... 8</td>
<td>Number of variables to be read: &quot;Addr1&quot; - &quot;Addr8&quot;</td>
</tr>
<tr>
<td>Addr1 - Addr8:</td>
<td>I</td>
<td>ANY</td>
<td>&quot;DBName&quot;.&lt;VarName&gt;</td>
<td>Variable identifiers from NC Var selector</td>
</tr>
<tr>
<td>Unit1 - Unit8:</td>
<td>I</td>
<td>BYTE</td>
<td>---</td>
<td>Area address, optional for variable addressing</td>
</tr>
<tr>
<td>Column1 - Column8:</td>
<td>I</td>
<td>WORD</td>
<td>---</td>
<td>Column address, optional for variable addressing</td>
</tr>
<tr>
<td>Line1 - Line8:</td>
<td>I</td>
<td>WORD</td>
<td>---</td>
<td>Line address, optional for variable addressing</td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: Negative acknowledgement of job or execution of job impossible</td>
</tr>
<tr>
<td>NDR:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: Job successfully executed Data is available</td>
</tr>
<tr>
<td>State:</td>
<td>O</td>
<td>WORD</td>
<td>---</td>
<td>See paragraph &quot;Error identifiers&quot;</td>
</tr>
<tr>
<td>RD1 - RD8:</td>
<td>I/O</td>
<td>ANY</td>
<td>P#Mm.n BYTE x ...</td>
<td>Target area for read data</td>
</tr>
</tbody>
</table>

Error identifiers

<table>
<thead>
<tr>
<th>NC variables</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>High byte</td>
<td>Low byte</td>
<td></td>
</tr>
<tr>
<td>1 - 8</td>
<td>1</td>
<td>Access error</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>Error in job</td>
</tr>
</tbody>
</table>
Drive-specific variables

If an error occurs while reading/writing a drive-specific variable (DB1200.DBX3000.1 == 1), an error number is displayed in the access result which is based on the error numbers defined in the PROFIdrive profile.

Determining the significance of the access result:
1. Calculation of the error number of the PROFIdrive profile
   \(<\text{Error number of PROFIdrive profile}> = \text{result of access} - 20_8, \text{ or } 36_8.\)
2. Determining the significance of the error number of the PROFIdrive profile
   The error numbers of the PROFIdrive profile are described in:

References
SINAMICS S120 Drive Functions Function Manual; Chapter "Communication" > "Communication according to PROFIdrive" > "Acyclic communication" > "Structure of orders and responses" > paragraph "Error values in parameter responses"

Configuration steps

Proceed as follows to read NC variables:
- Select variables with the NC VAR selector.
- Save selected variables in a *.VAR file.
- Generate a STEP 7 *.STL source file.
Generate a DB with the associated address data.

Enter the symbol for the generated DB in the symbol table so that it is possible to access the address parameters symbolically in the user program.

Parameterizing the FB2

Pulse diagram

Call example

Reading three channel-specific machine data from channel 1, whose address specifications are stored in DB120.

Specification of data

The data is selected with the NC VAR selector and saved in the DB120.VAR file. Then the DB120.AWL file is created from this.

S7 (ALIAS) names are selected.

To adopt the channel designation into the variable name and to delete the characters "[" and "]", which are not permitted in a STEP 7 symbol, new S7 names are selected:

<table>
<thead>
<tr>
<th>Area</th>
<th>Block</th>
<th>Name</th>
<th>Type</th>
<th>No.</th>
<th>Byte</th>
<th>S7 Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>C[1]</td>
<td>M</td>
<td>MD20070 $MC_AXCONF_MA-CHAX_USED[1]</td>
<td>CHAR</td>
<td>20070</td>
<td>1</td>
<td>C1AxConfMachAxUsed1</td>
</tr>
<tr>
<td>C[1]</td>
<td>M</td>
<td>MD20090 $MC_SPIND_DEF_MASTER_SPIND</td>
<td>INT</td>
<td>20090</td>
<td>1</td>
<td>C1SpindDefMasterSpind</td>
</tr>
</tbody>
</table>
S7 symbol table

"NCVAR" is entered in the S7 symbol table as a symbolic name for the data block DB120:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operand</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCVAR</td>
<td>DB120</td>
<td>DB120</td>
</tr>
</tbody>
</table>

File DB120.AWL must be compiled and transferred to the PLC.

Parameterization of FB2 with instance DB110:

```plaintext
DATA_BLOCK DB110 // Unassigned user DB, as instance for FB2
FB2
BEGIN
END_DATA_BLOCK
Function FC "VariablenCall" : VOID
  U  I 7.7; // Unassigned machine control panel key
  S  M 100.0; // Activate req.
  U  M 100.1; // NDR completed message
  R  M 100.0; // Terminate job
  U  I 7.6; // Manual error acknowledgment
  U  M 102.0; // Error pending
  R  M 100.0; // Terminate job
  CALL FB2, DB110(Req := M 100.0,
                   NumVar := 3, // Read three variables
                   Addr1 := "NCVAR".C1AxConfMachAxUsed1,
                   Addr2 := "NCVAR".C1AxConfMachAxUsed2,
                   Addr3 := "NCVAR".C1SpindDefMasterSpind,
                   Error := M102.0,
                   NDR := M100.1,
                   State := MW104,
                   RD1 := P#DB99.DBX0.0 BYTE 1,
                   RD2 := P#DB99.DBX1.0 BYTE 1,
                   RD3 := P#M110.0 INT 1);
```

Example: Variable addressing

Reading two R parameters from channel 1, whose address specifications are stored in DB120 as the basic type. The R parameter number is parameterized via parameter "Line<x>".

```plaintext
DATA_BLOCK DB120
VERSION : 0.0
STRUCT
```
Cl_RP_rpa0_0:
  STRUCT
  SYNTAX_ID : BYTE := B#16#82;
  area_and_unit : BYTE := B#16#41;
  column : WORD := W#16#1;
  line : WORD := W#16#0;
  block_type : BYTE := B#16#15;
  NO. OF LINES : BYTE := B#16#1;
  type : BYTE := B#16#F;
  length : BYTE := B#16#8;
  END_STRUCT;
END_STRUCT;
BEGIN
  END_DATA_BLOCK
  CALL FB2, DB110(
    Req := M 0.0,
    NumVar := 2,
    Addr1 := "NCVAR".Cl_RP_rpa0_0,
    Line1 := W#16#1,
    Addr2 := "NCVAR".Cl_RP_rpa0_0,
    Line2 := W#16#2,
    Error := M 1.0,
    NDR := M 1.1,
    State := MW 2,
    RD1 := P#M 4.0 REAL 1,
    RD2 := P#M 24.0 REAL 1);

**Classification of data types**

<table>
<thead>
<tr>
<th>NC data type</th>
<th>S7 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOUBLE</td>
<td>REAL</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>REAL2</td>
</tr>
<tr>
<td>FLOAT</td>
<td>REAL</td>
</tr>
<tr>
<td>LONG</td>
<td>DINT</td>
</tr>
<tr>
<td>INTEGER</td>
<td>DINT</td>
</tr>
<tr>
<td>UINT_32</td>
<td>DWORD</td>
</tr>
<tr>
<td>INT_16</td>
<td>INT</td>
</tr>
<tr>
<td>UINT_16</td>
<td>WORD</td>
</tr>
<tr>
<td>UNSIGNED</td>
<td>WORD</td>
</tr>
<tr>
<td>CHAR</td>
<td>CHAR or BYTE</td>
</tr>
<tr>
<td>STRING</td>
<td>STRING</td>
</tr>
<tr>
<td>BOOL</td>
<td>BOOL</td>
</tr>
<tr>
<td>DATETIME</td>
<td>DATE_AND_TIME</td>
</tr>
</tbody>
</table>
Example
For example, to be able to read an NC variable of the type DOUBLE without adapting the format, an ANY pointer with REAL2 type must be specified in the destination area "RDx" (e.g.: P#M100.0 REAL2). If the basic program recognizes REAL2 as the target type when reading a variable of the DOUBLE type, the data is transferred to the PLC data area as a 64-bit floating-point number.

13.17.3 FB3: PUT - write NC variables

Function
The FB3 "PUT" function block is used to write variables from the NC area. In order to reference the NC variables, they are first selected with the "NC VAR selector" tool and generated as STL source in a data block. A name must then be assigned to this data block in the S7 symbol table. When calling FB3, the variable addresses are transferred in the following form: Parameter "Addr1" to "Addr8" = "<DB name>".<S7 name>

Request for writing NC variables
Call of FB3 with positive edge change, parameter "Req" = 0 → 1
S7 names of the NC variables: Parameter "Addr1" to "Addr8" = "NCVAR".<S7 name>"  
Pointer for writing the variable values: Parameters "RD1" to "RD8" = "P#<Address>"

Completion of the write request
Write request successfully completed: Parameter "Done" == 1.
Write request with error completed: "Parameter "Error" == 1, error cause in parameter "State"

Prerequisites
- Release of the NC/PLC communication by OB100, FB1 parameter "NCKomm" = 1
- For the data block DB120 (data interface), the S7 Symbol Editor must be used to assign a symbol (default: NCVAR) in the S7 symbol table of the S7 project. Via this symbol, the NC variable is then specified in the FB3 parameters "Addr<x>", e.g. "ADDR1":= "NCVAR".<NC-Variable>"

General conditions
- FB3 has multi-instance capability.
- Every call of FB3 must be assigned a separate instance DB from the user area.
- In order to define machine data and GUD without a password, the protection level of the data you want to access must be redefined to the lowest level.

References:
- Commissioning Manual; Section: "Protection levels concept"
- Programming Manual, Job Planning; Section: "Define protection levels for user data"

- When channel-specific variables are written, only variables from exactly one channel may be addressed via "Addr1" to "Addr8" if FB2 is called.
When drive-specific variables are written, only variables from exactly one SERVO drive object may be addressed via "Addr1" to "Addr8" if FB2 is called. The SERVO drive object must be assigned to a machine axis of the NC. The line index corresponds to the logical drive number.

In a write job, only variables from the same area, channel, or drive object can be written.

**Note**

**Error case**

When writing variables from different channels or drive objects, or simultaneously from a channel and a drive object, an error message is output:

- "Error" == TRUE
- "State" == W#16#02

**Variable addressing**

For some NC variables, it is necessary to select "Area no." and/or "Line" or "Column" from the NC VAR selector. It is possible to select a basic type, i.e. "Area no.", "Line" and "Column" are preassigned "0". The values of the "Area no.", "Line" and "Column" specified by the NC VAR selector are checked for a "0" in FB2. If an NC-VAR selector value == "0", the corresponding value of the FB3 parameter is adopted. To do this, the FB3 parameters "Unit<x>", "Column<x>", and "Line<x>", with <x> = 1 - 8, must be written before FB3 is called.

<table>
<thead>
<tr>
<th>FB3 parameter</th>
<th>NC VAR selector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit&lt;x&gt;</td>
<td>Area no.</td>
</tr>
<tr>
<td>Column&lt;x&gt;</td>
<td>Column</td>
</tr>
<tr>
<td>Line&lt;x&gt;</td>
<td>Line</td>
</tr>
</tbody>
</table>

NCK variables within one group can be combined in a job:

<table>
<thead>
<tr>
<th>Group</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C[1]</td>
</tr>
<tr>
<td>2</td>
<td>C[2]</td>
</tr>
<tr>
<td>3</td>
<td>V[.]</td>
</tr>
</tbody>
</table>

The same rules apply for channels 3 to 10 as for group 1 and group 2 shown in the example.

**Note**

The number of usable variables can be less than eight when simultaneously writing several variables of the "String" type.

**Declaration of the function**

FUNCTION_BLOCK FB3
VAR_INPUT

P3: Basic PLC program for SINUMERIK 840D sl
13.17 Block descriptions
P3: Basic PLC program for SINUMERIK 840D sl

13.17 Block descriptions

Req :    BOOL;
NumVar :   INT;
Addr1 :    ANY;
Unit1 :    BYTE ;
Column1 :  WORD;
Line1 :    WORD;
Addr2 :    ANY;
Unit2 :    BYTE ;
Column2 :  WORD;
Line2 :    WORD;
Addr3 :    ANY;
Unit3 :    BYTE ;
Column3 :  WORD;
Line3 :    WORD;
Addr4 :    ANY;
Unit4 :    BYTE ;
Column4 :  WORD;
Line4 :    WORD;
Addr5 :    ANY;
Unit5 :    BYTE ;
Column5 :  WORD;
Line5 :    WORD;
Addr6 :    ANY;
Unit6 :    BYTE ;
Column6 :  WORD;
Line6 :    WORD;
Addr7 :    ANY;
Unit7 :    BYTE ;
Column7 :  WORD;
Line7 :    WORD;
Addr8 :    ANY;
Unit8 :    BYTE ;
Column8 :  WORD;
Line8 :    WORD;

END_VAR

VAR_OUTPUT
Error :      BOOL;
Done :       BOOL;
State :      WORD;

END_VAR

VAR_IN_OUT
Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req:</td>
<td>I</td>
<td>BOOL</td>
<td>-</td>
<td>Job start with positive signal edge</td>
</tr>
<tr>
<td>NumVar:</td>
<td>I</td>
<td>INT</td>
<td>1 ... 8</td>
<td>Number of variables to be written: &quot;Addr1&quot; - &quot;Addr8&quot;</td>
</tr>
<tr>
<td>Addr1 - Addr8:</td>
<td>I</td>
<td>ANY</td>
<td>&quot;DBName&quot;.&lt;VarName&gt;</td>
<td>Variable identifiers from NC Var selector</td>
</tr>
<tr>
<td>Unit1 - Unit8:</td>
<td>I</td>
<td>BYTE</td>
<td>-</td>
<td>Area address, optional for variable addressing</td>
</tr>
<tr>
<td>Column1 - Column8:</td>
<td>I</td>
<td>WORD</td>
<td>-</td>
<td>Column address, optional for variable addressing</td>
</tr>
<tr>
<td>Line1 - Line8:</td>
<td>I</td>
<td>WORD</td>
<td>-</td>
<td>Line address, optional for variable addressing</td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Negative acknowledgement of job or execution of job impossible</td>
</tr>
<tr>
<td>Done:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Job successfully executed</td>
</tr>
<tr>
<td>State:</td>
<td>O</td>
<td>WORD</td>
<td>-</td>
<td>See paragraph &quot;Error identifiers&quot;</td>
</tr>
<tr>
<td>SD1 - SD8:</td>
<td>I/O</td>
<td>ANY</td>
<td>P#Mm.n BYTE x... P#DBnr.dbx.m.n BYTE x</td>
<td>Data to be written</td>
</tr>
</tbody>
</table>

Error identifiers

NC variables

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>High byte</td>
<td>Low byte</td>
<td></td>
</tr>
<tr>
<td>1 - 8</td>
<td>1</td>
<td>Access error</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>Error in job</td>
</tr>
</tbody>
</table>
| 0      | 3         | Negative acknowledgement, job not executable | Internal error, try:  
|        |           |                                           | ● Check job  
|        |           |                                           | ● NC reset |
| 1 - 8  | 4         | Data areas or data types do not match or string is empty | Check data in "SD1" - "SD8" |
### Drive-specific variables

If an error occurs while reading/writing a drive-specific variable (DB1200.DBX3000.1 == 1), an error number is displayed in the access result which is based on the error numbers defined in the PROFIdrive profile.

#### Configuration steps

To write NC variables, the same configuration steps are required as for reading NC variables. It is useful to store the address data of all NC variables to be read or written in a DB.
Pulse diagram

1. User: Set request, Req = 0 \rightarrow 1
2. FB4: PI service successfully completed, Done = 1
   User: Reset request, IF Done == 1 THEN Req = 0
3. User: IF Done == 1 THEN reset request: 1 \rightarrow 0
4. FB4: Reset job confirmation, Done = 0
5. User: IF Done == 0 AND Error == 0 THEN reset request Req = 1 \rightarrow 0 \textbf{not permissible}
6. FB4: PI service completed with errors, Error = 1
   User: Reset request, IF Done == 1 OR Error == 1 THEN Req = 0, possible further error handling

Call example

Writing of three channel-specific machine data of channel 1:

Selection of three data with NC VAR selector and storage in file DB120.VAR

S7 (ALIAS) names are selected in order to adopt the block designation into the name and to remove the characters [], which are not permitted in a STEP 7 symbol.

<table>
<thead>
<tr>
<th>Area</th>
<th>Block</th>
<th>Name</th>
<th>Type</th>
<th>Byte</th>
<th>S7 Name</th>
</tr>
</thead>
</table>

Entry NCVAR for DB120 with the S7 SYMBOL Editor

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operand</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCVAR</td>
<td>DB120</td>
<td>DB120</td>
</tr>
</tbody>
</table>

File DB120.AWL must be compiled and transferred to the PLC.

Calling and parameterizing the FB3 with instance DB111

```
DATA_BLOCK DB111    // Unassigned user DB, as instance for FB3
FB3
BEGIN
Function FC "VariablenCall" : VOID
```
Example: Variable addressing

Writing two R parameters from channel 1, whose address specifications are stored in DB120 as the basic type. The R parameter number is parameterized via parameter LineX.

DATA_BLOCK DB120
VERSION : 0.0
STRUCT
C1_RP_rpa0_0:
    STRUCT
        SYNTAX_ID : BYTE := B#16#82;
        area_and_unit : BYTE := B#16#41;
        column : WORD := W#16#1;
        line : WORD := W#16#0;
        block type : BYTE := B#16#15;
        NO. OF LINES : BYTE := B#16#1;
        type : BYTE := B#16#F;
        length : BYTE := B#16#8;
    END_STRUCT;
END_STRUCT;
BEGIN
END_DATA_BLOCK
CALL FB3, DB122(
    Req := M 100.0,
    NumVar := 3, // Write three variables
    Addr1 := "NCVAR".rpa_5C1RP,
    Addr2 := "NCVAR".rpa_11C1RP,
    Addr3 := "NCVAR".rpa_14C1RP,
    Error := M102.0,
    Done := M100.1,
    State := MW104,
    SD1 := P#DB99.DBX0.0 REAL 1,
    SD2 := P#DB99.DBX4.0 REAL 1,
    SD3 := P#M110.0 REAL 1);
Req := M 10.0,
NumVar := 2,
Addr1 := "NCVAR".C1_RP_rp0_0,
Line1 := W#16#1,
Addr2 := "NCVAR".C1_RP_rp0_0,
Line3 := W#16#2
Error := M 11.0,
Done := M 11.1,
State := MW 12,
SD1 := P#M 4.0 REAL 1,
SD2 := P#M 24.0 REAL 1);

13.17.4 FB4: Request PI service

13.17.4.1 FB4: PI_SERV - request PI service

Function

The function block FB4 "PI_SERV" is used to start PI services.
The available PI services are described in the following chapters with their specific parameters.
An overview of the available PI services can be found in: List of available PI services (Page 941)

Note

Due to the large number of "WVar" parameters, it is recommended that you use the function
block FB7 instead of FB4. See Chapter "FB7: PI_SERV2 - request PI service (Page 973)".

Start of a PI service

Request to start a PI service: Call of FB4 with positive edge change, parameter "Req" = 0 → 1

Completion of a PI service

Job or PI service successfully completed: Parameter "Done" == 1.
Job or PI service completed with error: "Parameter "Error" == 1, error cause in parameter "State"

Prerequisites

- Release of the NC/PLC communication by OB100, FB1 parameter "NCKomm" = 1
- For the data block DB16 (data interface of the PI services), the S7 Symbol Editor must be
  used to assign a symbol (default: PI) in the S7 symbol table of the S7 project. The requested
  PI service is then specified via this symbol in the FB4 parameter "PIService", e.g.
  "PIService:= "PI"."<PI service>"
General conditions

- Every call of FB4 must be assigned a separate instance DB from the user area.
- The start of a PI service (FB4 call with "Req" = 1) is only permitted in the cyclic part of the PLC basic program (OB1).
  If the PI service is not started (FB4 call with "Req" = 0), the parameters can also be written in the start-up part of the PLC basic program (OB100). The PI service can then be started using the already written parameters in the cyclic part of the PLC basic program (OB1) by calling FB4 with "Req" = 1.
- The execution of the PI service generally extends over several PLC cycles.

Declaration of the function

FUNCTION_BLOCK FB4
VAR_INPUT
  w  Req : BOOL;
  PIService : ANY;
  Unit : INT;
  Addr1 : ANY;
  Addr2 : ANY;
  Addr3 : ANY;
  Addr4 : ANY;
  WVar1 : WORD;
  WVar2 : WORD;
  WVar3 : WORD;
  WVar4 : WORD;
  WVar5 : WORD;
  WVar6 : WORD;
  WVar7 : WORD;
  WVar8 : WORD;
  WVar9 : WORD;
  WVar10 : WORD;
END_VAR
VAR_OUTPUT
  Error : BOOL;
  Done : BOOL;
  State : WORD;
END_VAR
Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Ty pe</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Job request</td>
</tr>
<tr>
<td>PIService:</td>
<td>I</td>
<td>ANY</td>
<td>&quot;&lt;DBName&gt;&quot;.&quot;&lt;PI service&gt;&quot;</td>
<td>Requested PI service</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• &lt;DBName&gt;: symbol name for DB16, default: &quot;PI&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• &lt;PI service&gt;: List of available PI services (Page 941)</td>
</tr>
<tr>
<td>Unit:</td>
<td>I</td>
<td>INT</td>
<td>1, 2, 3 ...</td>
<td>Area number</td>
</tr>
<tr>
<td>Addr1 to Addr4:</td>
<td>I</td>
<td>ANY</td>
<td>&quot;&lt;DBName&gt;&quot;.&quot;&lt;VarName&gt;&quot;</td>
<td>Reference to a string</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Significance depending on selected PI service</td>
</tr>
<tr>
<td>WVar1 to WVar10:</td>
<td>I</td>
<td>WORD</td>
<td>-32768 ... 32767 8000H - 7FFFH</td>
<td>INTEGER or WORD variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Significance depending on selected PI service</td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Error status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Negative acknowledgment of job or execution of job impossible</td>
</tr>
<tr>
<td>Done:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Job status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Job successfully executed</td>
</tr>
<tr>
<td>State:</td>
<td>O</td>
<td>WORD</td>
<td>1)</td>
<td>Error detection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Only relevant if &quot;Error&quot; == 1</td>
</tr>
</tbody>
</table>

1) State Description Note

- **3** Negative acknowledgement, job not executable
  - Internal error, possible remedy through an NC RESET
- **6** FIFO full
  - Repeat the command
- **7** Option not set
  - FB1 parameter "NCKomm" is not set
- **9** Transmission occupied
  - Repeat the command
- **13** Addr1..Addr4: Reference invalid
  - Specify missing string
- **14** PI service is unknown
  - The PI service specified in the "PIService" parameter is unknown. → Check notation.
- **15** Addr1..Addr4: String too long
  - Check string lengths

Call example

**Function: Program selection in channel 1 (main program and workpiece program)**

Entry of PI service for DB16 and STR for DB124 using the S7 SYMBOL editor:

<table>
<thead>
<tr>
<th>Parameterization</th>
<th>Symbol</th>
<th>Operand</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PI</td>
<td>DB16</td>
<td>DB16</td>
</tr>
<tr>
<td></td>
<td>STR</td>
<td>DB124</td>
<td>DB124</td>
</tr>
</tbody>
</table>
DATA_BLOCK DB126  // Instance for FB4, unassigned user DB
FB4
BEGIN
END_DATA_BLOCK
// ------------------------------------------------------------------------
DATA_BLOCK DB124
STRUCT
  PName:        string[32]:= '_N_TEST_MPF  // Main program
                 ';
  Path:         string[32]:= '/
                 _N_MPF_DIR/';  // Main program path
  PName_WST:    string[32]:= '_N_ABC_MPF';  // Workpiece program
  Path_WST:     string[32]:= '/_N_WCS_DIR/
                 _N_ZYL_WPD';  // Workpiece program path
END_STRUCT
BEGIN
END_DATA_BLOCK
// ------------------------------------------------------------------------
Function FC "PICall" : VOID
  U I 7.7;   // Unassigned machine control panel key
  S M 0.0;   // Activate req.
  U M 1.1;   // Done completed message
  R M 0.0;   // Terminate job
  U I 7.6;   // Manual error acknowledgment
  U M 1.0;   // Error pending
  R M 0.0;   // Terminate job

CALL FB4, DB126 (  
  Req:= M0.0,
  PIService:= "PI".SELECT,     // PI service: SELECT
  Unit:= 1,                    // Channel 1
  Addr1:= "STR".Path,         // Main program: Path
  Addr2:= "STR".PName,        // Main program: Program
  // Addr1:= "STR".Path_WST,   // Workpiece: Path
  // Addr2:= "STR".PName_WST,  // Workpiece: Program
  Error:= M1.0,
  Done:= M1.1,
  State:= MW2
);
Flow diagram

1. User: Set request, Req = 0 → 1
2. FB4: PI service successfully completed, Done = 1
   User: Reset request, IF Done == 1 THEN Req = 0
3. User: IF Done == 1 THEN reset request: 1 → 0
4. FB4: Reset job confirmation, Done = 0
5. User: IF Done == 0 AND Error == 0 THEN reset request Req = 1 → 0 not permissible
6. FB4: PI service completed with errors, Error = 1
   User: Reset request, IF Done == 1 OR Error == 1 THEN Req = 0, possible further error handling

13.17.4.2 List of available PI services

**General PI services**

<table>
<thead>
<tr>
<th>PI service</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASUB (Page 942)</td>
<td>Assign interrupt</td>
</tr>
<tr>
<td>CANCEL (Page 943)</td>
<td>Execute cancel</td>
</tr>
<tr>
<td>CONFIG (Page 943)</td>
<td>Reconfiguration of tagged machine data</td>
</tr>
<tr>
<td>DIGION (Page 944)</td>
<td>Digitizing on</td>
</tr>
<tr>
<td>DIGIOF (Page 944)</td>
<td>Digitizing off</td>
</tr>
<tr>
<td>FINDBL (Page 944)</td>
<td>Activate block search</td>
</tr>
<tr>
<td>LOGIN (Page 945)</td>
<td>Activate password</td>
</tr>
<tr>
<td>LOGOUT (Page 945)</td>
<td>Reset password</td>
</tr>
<tr>
<td>NCRES (Page 945)</td>
<td>Trigger NC-RESET</td>
</tr>
<tr>
<td>SELECT (Page 945)</td>
<td>Select program for processing for one channel</td>
</tr>
<tr>
<td>SETUDT (Page 946)</td>
<td>Sets the current user data to active</td>
</tr>
<tr>
<td>SETUFR (Page 947)</td>
<td>Activate user frame</td>
</tr>
<tr>
<td>RETRAC (Page 947)</td>
<td>Retraction of the tool in the tool direction</td>
</tr>
</tbody>
</table>

**PI services of tool management**

<table>
<thead>
<tr>
<th>PI service</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRCEDN (Page 948)</td>
<td>Create a tool cutting edge with specification of the T number</td>
</tr>
<tr>
<td>CREACE (Page 948)</td>
<td>Create a tool cutting edge with the next higher/free D number</td>
</tr>
</tbody>
</table>
13.17.4.3 PI service: ASUB

Function: Assign interrupt

A program stored on the NCK is assigned an interrupt signal of a channel. The program must be executable and the path and program name must be specified completely and correctly. For detailed information, please refer to:

References

Programming Manual, Job Planning; Section: "File and Program Management" > "Program Memory".

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;, ASUP</td>
<td>Assign interrupt 1)</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>Channel</td>
</tr>
<tr>
<td>Addr1:</td>
<td>STRING</td>
<td>Path name</td>
<td></td>
</tr>
<tr>
<td>Addr2:</td>
<td>STRING</td>
<td>Program name</td>
<td></td>
</tr>
<tr>
<td>WVar1:</td>
<td>WORD</td>
<td>1, 2, 3 ... 8</td>
<td>Interrupt number</td>
</tr>
<tr>
<td>WVar2:</td>
<td>WORD</td>
<td>1, 2, 3 ... 8</td>
<td>Priority</td>
</tr>
<tr>
<td>WVar3:</td>
<td>WORD</td>
<td>0, 1</td>
<td>LIFTFAST 2)</td>
</tr>
</tbody>
</table>
13.17.4.4 PI service: CANCEL

Function: Execute cancel

Triggers the "Cancel" function equivalent to the corresponding "Cancel alarm" button on the user interface (operator panel front).

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.CANCEL</td>
<td>Cancel</td>
</tr>
</tbody>
</table>

13.17.4.5 PI service: CONFIG

Function: Reconfiguration

The reconfiguration command activates machine data which has been entered sequentially by the operator or the PLC, almost in parallel.

The command can only be activated when the controller is in RESET state or the program is interrupted (NC stop at block limit). An FB4 error checkback signal is output if this condition is not fulfilled (state = 3).

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.CONFIG</td>
<td>Reconfiguration</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td>1</td>
<td>Classification</td>
</tr>
</tbody>
</table>
13.17.4.6 PI service: DIGION

Function: Digitizing on
Selecting digitizing in the parameterized channel.

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.DIGION</td>
<td>Digitizing on</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>Channel</td>
</tr>
</tbody>
</table>

13.17.4.7 PI service: DIGIOF

Function: Digitizing off
Deactivating digitizing in the parameterized channel.

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.DIGIOF</td>
<td>Digitizing off</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>Channel</td>
</tr>
</tbody>
</table>

13.17.4.8 PI service: FINDBL

Function: Activate block search
A channel is switched to search mode and the appropriate acknowledgement then transmitted. The search is then executed immediately by the NC. The search pointer must already be in the NC at this point in time. The search can be interrupted at any time by an NC RESET. Once the search is successfully completed, the normal processing mode is reactivated automatically. NC start then takes effect from the located search target.

It is the sole responsibility of the operator to ensure a collision-free approach path.

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.FINDBL</td>
<td>block search</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1 to 10</td>
<td>Channel</td>
</tr>
<tr>
<td>WVar1:</td>
<td>WORD</td>
<td>1, 2, 3</td>
<td>Preprocessing mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Without calculation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: With calculation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3: With main block consideration</td>
</tr>
</tbody>
</table>
13.17.4.9 PI service: LOGIN

Function: Create password
Transfers the parameterized password to the NC. The passwords generally consist of eight characters. If required, blanks must be added to the string of the password.

Example
Password: STRING[8] := 'SUNRISE';

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.LOGIN</td>
<td>Create password</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1</td>
<td>NC</td>
</tr>
<tr>
<td>Addr1:</td>
<td>STRING</td>
<td>8 characters</td>
<td>Password</td>
</tr>
</tbody>
</table>

13.17.4.10 PI service: LOGOUT

Function: Reset password
The password last transferred to the NC is reset.

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.LOGOUT</td>
<td>Reset password</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1</td>
<td>NC</td>
</tr>
</tbody>
</table>

13.17.4.11 PI service: NCRES

Function: Trigger NC-RESET
Triggers an NC-RESET. Parameters "Unit" and "WVar1" must always be set to 0.

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.NCRES</td>
<td>Trigger NC-RESET</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>WVar1:</td>
<td>WORD</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

13.17.4.12 PI service: SELECT

Function: Select processing for a channel
A program stored on the NC is selected for one channel for execution. This is possible only if the file may be executed. The path names and the program names are to be written in correct notation. For detailed information, please refer to:
References
Programming Manual, Job Planning; Section: "File and Program Management" > "Program Memory".

Possible block types

<table>
<thead>
<tr>
<th>Block types</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Workpiece directory</td>
<td>WPD</td>
</tr>
<tr>
<td>Main program</td>
<td>MPF</td>
</tr>
<tr>
<td>Subprogram</td>
<td>SPF</td>
</tr>
<tr>
<td>Cycles</td>
<td>CYC</td>
</tr>
<tr>
<td>Asynchronous subprograms</td>
<td>ASP</td>
</tr>
<tr>
<td>Binary files</td>
<td>BIN</td>
</tr>
</tbody>
</table>

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.SELECT</td>
<td>Program selection</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>Channel</td>
</tr>
<tr>
<td>Addr1:</td>
<td>STRING</td>
<td>-</td>
<td>Path name</td>
</tr>
<tr>
<td>Addr2:</td>
<td>STRING</td>
<td>-</td>
<td>Program name</td>
</tr>
</tbody>
</table>

13.17.4.13 PI service: SETUDT

Function: Set current user data active

The current user data, such as tool offsets, basic frames and settable frames are set to active in the next NC block (only in stop state).

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.SETUDT</td>
<td>Activate user data</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>Channel</td>
</tr>
<tr>
<td>WVar1:</td>
<td>WORD</td>
<td>1, 2, 3, 4, 5</td>
<td>User Data Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Active tool offset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: Active basic frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3: Active settable frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4: Active global basic frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5: Active global settable frame</td>
</tr>
<tr>
<td>WVar2:</td>
<td>WORD</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>WVar3:</td>
<td>WORD</td>
<td>0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
13.17.4.14 PI service: SETUFN

Function: Activate user frames

User frames are loaded to the NC. All necessary frame values must be transferred to the NC first with FB3 "Write variables".

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.SETUFN</td>
<td>Activate user frames</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>Channel</td>
</tr>
</tbody>
</table>

13.17.4.15 PI service: RETRAC

Function: Select JOG retract

Selects the JOG retract mode. The retraction axis, i.e. the geometry axis, with which the retraction is executed can be determined by the NC automatically or specified explicitly.

The mode remains active until it is ended with RESET.

Note

The RETRAC PI service can only be activated in JOG mode in the "Reset" state.

Automatic determination

For automatic determination, the geometric axis is selected as a retraction axis, which is perpendicular (orthogonal) to the currently selected working plane:

- G17: Retraction axis ⇒ 3rd geometry axis (Z)
- G18: Retraction axis ⇒ 2nd geometry axis (Y)
- G19: Retraction axis ⇒ 1st geometry axis (X)

Note

OPI variable retractState

The active retraction axis can be read via the OPI variable retractState.Bit 2/3.

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.RETRAC</td>
<td>Select JOG retract mode</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>Channel</td>
</tr>
<tr>
<td>Signal</td>
<td>Type</td>
<td>Value range</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WVar1:</td>
<td>WORD</td>
<td>0, 1, 2, 3</td>
<td>Retraction axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Automatic determination of the retraction axis by the NC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Retraction axis is the 1st geometry axis of the WCS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: Retraction axis is the 2nd geometry axis of the WCS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3: Retraction axis is the 3rd geometry axis of the WCS</td>
</tr>
<tr>
<td>WVar2:</td>
<td>WORD</td>
<td>0</td>
<td>Reserved. The value must be pre-assigned with 0.</td>
</tr>
</tbody>
</table>

### 13.17.4.16 PI service: CRCEDN

**Function:** Creates new cutting edge

If the T number of an existing tool is entered in parameter “T Number” in the PI service, then a tool edge for the existing tool is created (in this case, the parameter “D number”, i.e. the number of the edge to be created, has a value range of 1, 2, 3 ... 9.

If a positive T number is specified as a parameter and the tool for the T number entered does not exist, the PI service is aborted.

If a value of 0 is entered as the T number (model of absolute D numbers), the D number values can range from 1, 2, 3 to -31999. The new cutting edge is set up with the specified D number.

If the specified cutting edge already exists, the PI service is aborted in both cases.

**Description of formal parameters**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;,CRCEDN</td>
<td>Create new cutting edge</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td>T number of tool for which cutting edge must be created. A setting of 00000 states that the cutting edge should not refer to any particular tool (absolute D number).</td>
<td></td>
</tr>
<tr>
<td>WVar2:</td>
<td>INT</td>
<td>1 ... 9</td>
<td>Edge number of tool cutting edge</td>
</tr>
</tbody>
</table>

### 13.17.4.17 PI service: CREACE

**Function:** Create tool cutting edge

Creation of the cutting edge with the next higher / next unassigned D number for the tool with the transferred T number in TO, TS (if present). The cutting edge for the OEM cutting edge data is set up simultaneously in the TUE block (if one is present).
Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.CREACE</td>
<td>Create tool cutting edge</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td>T number</td>
<td></td>
</tr>
</tbody>
</table>

### 13.17.4.18 PI service: CREATEO

**Function: Create tool**

Creation of a tool with specification of a T number. The tool is entered as existing in the tool directory area (TV). The first "cutting edge" D1 (with zero contents) is created for tool offsets in the TO block. D1 (with zero contents) is also created for the OEM "cutting edge" data in the TUE block - if one is present. If a TU block exists, it will contain the data set for the tool.

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.CREATEO</td>
<td>Create tool</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td>T number</td>
<td></td>
</tr>
</tbody>
</table>

### 13.17.4.19 PI service: DELETEC

**Function: Delete a tool cutting edge**

If the T number of an existing tool is specified in parameter "T number" in the PI service, then a cutting edge is deleted for this particular tool (in this case, parameter "D number" (number of cutting edge to be created) has a value range of 00001 - 00009). If a positive T number is specified as a parameter and the tool for the T number entered does not exist, then the PI service is aborted. If a value of 00000 is entered as the T number (model of absolute D numbers), then the D number values can range from 00001 - 31999. If the specified cutting edge does not exist, then the PI service is aborted in both cases.

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.DELETEC</td>
<td>Delete cutting edge</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td>T number</td>
<td>T number of tool for which cutting edge must be cre-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ated. A setting of 00000 states that the cutting edge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>should not refer to any particular tool (absolute D</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>number).</td>
</tr>
<tr>
<td>WVar2:</td>
<td>INT</td>
<td>1 ... 9</td>
<td>Edge number of cutting edge that must be deleted</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - 31999</td>
<td></td>
</tr>
</tbody>
</table>
13.17.4.20 PI service: DELETO

Function: Delete tool

Deletes the tool assigned to the transferred T number with all cutting edges (in TO, in some cases TU, TUE and TG (type 4xx), TD and TS blocks).

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;,DELETO</td>
<td>Delete tool</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td></td>
<td>T number</td>
</tr>
</tbody>
</table>

13.17.4.21 PI service: MMCSEM

Function

The PI service is used to test/set and reset channel-specific semaphores from HMI and PLC. 10 semaphores are available per channel for protecting critical data areas.

Functions (PI services) are assigned to semaphores 1 to 6. The semaphores 7 to 10 can be freely used.

FB4 return values

- "Done" == TRUE
  The semaphore was set, the critical function can be called.
- "Error" == TRUE AND "State" == 3
  The semaphore has already been set, the critical function cannot be called at the present time.

Schematic sequence for free semaphore

Testing and setting the semaphore
IF semaphore == FREE
THEN
  Writing/reading of critical data
  Resetting the semaphore
ELSE // Semaphore is blocked
...
ENDIF

NOTICE

Resetting the semaphore

After the blocking of the critical data areas by setting the semaphore and the subsequent reading/writing of the data, it is essential that the critical data area is released again by resetting the semaphore.

Schematic sequence for blocked semaphore

Testing and setting the semaphore
IF semaphore == FREE
THEN
    ...
ELSE // Semaphore is blocked
    Set bit memory for "Function could not be executed, repeat necessary"
ENDIF

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.MMCSEM</td>
<td>Edit semaphore</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>Channel</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td></td>
<td>PI service-specific number of the semaphore</td>
</tr>
<tr>
<td>WVar2:</td>
<td>WORD</td>
<td>0, 1</td>
<td>Job type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Reset semaphore</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Test and set semaphore</td>
</tr>
</tbody>
</table>

1) Number PI service

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TMCRTO (create tool)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>TMFDPL (search for empty location for loading)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>TMMVTL (prepare magazine location for loading, unload tool)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>TMFPBP (search for location)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TMGETT (search for tool number)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>TSEARC (search for tool)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Freely usable</td>
<td></td>
</tr>
</tbody>
</table>

13.17.422 PI service: TMCRTO

Function: Create tool

Creation of a tool by specifying a name, a duplo number, e.g. with $TC_TP1[y] = Duplo number or $TC_TP2[y] = "Tool name". Or optionally using a T number, e.g. with y = T number

The tool is entered in the TV area (tool directory) as available.

The first cutting edge "D1" (with zero content) is created in the TO block for corrections.

The first cutting edge "D1" (with zero content) is created in the TS block for monitoring data.

In the TUE block, if it is available, the cutting edge "D1" is created for the OEM cutting edge data.

The TD block contains identifiers, duplo numbers, and the number of cutting edges (=1) for the T number that is optionally specified or assigned by the NC.

If a TU block exists, it will contain the data block for the tool.
After execution of the PI service, the T number of the tool created is available in the TV block under TnumWZV.

Note
Before and after this PI service, the PI service MMCSEM with parameter "WVar1" must be called with function number 1 for TMCRTO. See Section "PI service: MMCSEM (Page 950)".

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.TMCRTO</td>
<td>Create tool</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td>1)</td>
<td>T number</td>
</tr>
<tr>
<td>WVar2:</td>
<td>INT</td>
<td></td>
<td>Duplo number</td>
</tr>
<tr>
<td>Addr1:</td>
<td>STRING</td>
<td>Max. 32 characters</td>
<td>Tool name</td>
</tr>
</tbody>
</table>

1) T number > 0 ⇒ the T number is specified
   T number > -1 ⇒ the NC assigns the T number

13.17.4.23 PI service: TMFDPL

Function
Search for empty location for loading, depending on the parameter assignment:

- Location_number_to = -1 AND Magazine_number_to = -1
  Searches all magazines in the specified area (= channel) for an empty location for the tool specified with a T number. After execution of the PI service, the magazine and locations numbers found during the search are listed in the configuration block of the channel (component magCMCmdPar1 (magazine number) and magCMCmdPar2 (location number)). Location_number_ID and magazine_number_ID can be set as search criteria or not (= -1). The PI service is positively or negatively acknowledged depending on the search result.

- Location_number_to = -1 AND Magazine_number_to = Magazine_number
  An empty location for the tool specified with a T number is searched for in the specified magazine. Location_number_ID and magazine_number_ID can be set as a search criterion or with -1. The PI service is positively or negatively acknowledged depending on the search result.

- Location_number_to = Location_number AND Magazine_number_to = Magazine_number
  The specified location is checked, to confirm that it is free to be loaded with the specified tool. Location_number_ID and magazine_number_ID can be set as search criteria or -1. The PI service is acknowledged positively or negatively depending on the search result.
The parameters "WVar1" and "WVar2" are located at the source.

Loading: If the source is an internal loading magazine, then the parameters are located at the target (a real magazine).

Unloading: Source is always a real magazine.

**Note**

Before and after this PI service, the PI service MMCSEM with parameter "WVar1" must be called with function number 2 for TMFDPL. See Chapter "PI service: MMCSEM (Page 950)".

**Description of formal parameters**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.TMFDPL</td>
<td>Empty location for loading</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td>T number</td>
<td></td>
</tr>
<tr>
<td>WVar2:</td>
<td>INT</td>
<td>Location_number_where_to</td>
<td></td>
</tr>
<tr>
<td>WVar3:</td>
<td>INT</td>
<td>Magazine_number_to</td>
<td></td>
</tr>
<tr>
<td>WVar4:</td>
<td>INT</td>
<td>Location_number_ID</td>
<td></td>
</tr>
<tr>
<td>WVar5:</td>
<td>INT</td>
<td>Magazine_number_ID</td>
<td></td>
</tr>
</tbody>
</table>

**13.17.4.24 PI service: TMFPBP**

**Function: Empty location search**

The PI service searches the specified magazine(s) for an empty location which meets the specified criteria such as tool size and location type.

If the search is successful, the result can be read from the following OPI variables:

- magCMCmdPar1 (magazine number)
- magCMCmdPar2 (location number)

**Note**

The PI service can only be requested using FB7. See Chapter "FB7: PI_SERVICE - request PI service (Page 973)".

**Note**

Before and after this PI service, the PI service MMCSEM with parameter "WVar1" must be called with function number 4 for TMFPBP. See Chapter "PI service: MMCSEM (Page 950)".

**Description of formal parameters**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.TMFPBP</td>
<td>Empty location search</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>Signal</td>
<td>Type</td>
<td>Value range</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td></td>
<td>Magazine number of the magazine from which the search is to be performed</td>
</tr>
<tr>
<td>WVar2:</td>
<td>INT</td>
<td></td>
<td>Location number of the location in the magazine from &quot;WVar1&quot;</td>
</tr>
<tr>
<td>WVar3:</td>
<td>INT</td>
<td></td>
<td>Magazine number of the magazine up to which the search is to be performed</td>
</tr>
<tr>
<td>WVar4:</td>
<td>INT</td>
<td></td>
<td>Location number of the location in the magazine from &quot;WVar3&quot;</td>
</tr>
<tr>
<td>WVar5:</td>
<td>INT</td>
<td></td>
<td>Magazine number reference</td>
</tr>
<tr>
<td>WVar6:</td>
<td>INT</td>
<td></td>
<td>Location number reference</td>
</tr>
<tr>
<td>WVar7:</td>
<td>INT</td>
<td>0, 1, 2 ... 7</td>
<td>Number of required half locations to left</td>
</tr>
<tr>
<td>WVar8:</td>
<td>INT</td>
<td>0, 1, 2 ... 7</td>
<td>Number of required half locations to right</td>
</tr>
<tr>
<td>WVar9:</td>
<td>INT</td>
<td>0, 1, 2 ... 7</td>
<td>Number of required half locations in upward direction</td>
</tr>
<tr>
<td>WVar10:</td>
<td>INT</td>
<td>0, 1, 2 ... 7</td>
<td>Number of required half locations in downward direction</td>
</tr>
<tr>
<td>WVar11:</td>
<td>INT</td>
<td></td>
<td>Number of required location type</td>
</tr>
<tr>
<td>WVar12:</td>
<td>INT</td>
<td>0, 1, 2, 3, 4</td>
<td>Specifies the required search direction</td>
</tr>
</tbody>
</table>

Examples: Setting the search range

<table>
<thead>
<tr>
<th>From Location</th>
<th>To Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>WVar1</td>
<td>WVar2</td>
</tr>
<tr>
<td>#M1</td>
<td>#P1</td>
</tr>
<tr>
<td>#M1</td>
<td>#P1</td>
</tr>
<tr>
<td>#M1</td>
<td>-1</td>
</tr>
<tr>
<td>#M1</td>
<td>-1</td>
</tr>
<tr>
<td>#M1</td>
<td>#P1</td>
</tr>
<tr>
<td>#M1</td>
<td>#P1</td>
</tr>
<tr>
<td>#M1</td>
<td>#P1</td>
</tr>
<tr>
<td>#M1</td>
<td>-1</td>
</tr>
<tr>
<td>#M1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

P3: Basic PLC program for SINUMERIK 840D sl
13.17 Block descriptions
13.17.4.25 PI service: TMGETT

**Function:** Determine T number for the specified tool name with duplo number

The PI service is used to determine the T number of a tool via the tool name and duplo number. The result is written in BTSS variables in the TF block (parameterization, return parameters from TMGETT, TSEARC):

- resultNrOfTools
  - resultNrOfTools == 0: The specified tool was not found
  - resultNrOfTools == 1: The specified tool was found
- resultToolNr: T number of the specified tool with "resultNrOfTools" == 1

**Note**
Before and after this PI service, the PI service MMCSEM with parameter "WVar1" must be called with function number 5 for TMGETT. See Section "PI service: MMCSEM (Page 950)".

**Description of formal parameters**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.TMGETT</td>
<td>Determining the T number</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>Addr1:</td>
<td>STRING</td>
<td>Max. 32 characters</td>
<td>Tool name</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td></td>
<td>Duplo number</td>
</tr>
</tbody>
</table>

13.17.4.26 PI service: TMMVTL

**Function:** Prepare magazine location for loading, unload tool

The PI service is used to load, unload, and relocate tools:

1. Loading and unloading: loading point ↔ magazine)
2. Loading and unloading: loading point ↔ buffer storage, e.g. spindle
3. Relocation within a magazine
4. Relocation between different magazines
5. Relocation between magazine and buffer storage
6. Relocation within buffer storage

Case 1, 3, 4 and 5: The following BTSS variables of the block TM (magazine data: general data) are written:

- magCmd (area no. = TO unit, line = magazine number)
- magCmdState ← "acknowledgment"
Case 2 and 6: The following BTSS variables of the block TMC (magazine data: configuration data) are written:

- `magCBCmd` (area no. = TO unit)
- `magCBCmdState ← "acknowledgment"

**Loading**

- WVar2 "Location number_From", WVar3 "Magazine number_From"
  The tool location of the magazine is moved to the loading station/location for loading and the tool is loaded.

- WVar4 "Location number_To" == -1
  First, an empty location for the tool is searched for in the magazine. Then the empty location of the magazine is moved to the loading station/location for loading and the tool is loaded. After the PI service is carried out, the empty location number that is found is located in "magCMCmdPar2" (BTSS block TM), real magazine of the channel.

- WVar4 "Location number_To" == -2
  The tool is loaded into the current tool location of the magazine. After the PI service is carried out, the location number is located in "magCMCmdPar2" (BTSS block TM), real magazine of the channel.

**Unloading**

- WVar4 "Location number_To", WVar5 "Magazine number_To"
  The tool location of the magazine is moved to the loading station/location for loading and the tool is unloaded.

In the BTSS block TP (magazine data: location data), the magazine location of the removed tool is designated as free.

**Addressing the tool**

The tool can be addressed either using a T number or by means of the location and magazine numbers. The value -1 is assigned to an unused parameter.

**Note**

Before and after this PI service, the PI service MMCSEM with parameter "WVar1" must be called with function number 3 for TMMVTL. See Chapter "PI service: MMCSEM (Page 950)".

**Description of formal parameters**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.TMMVTL</td>
<td>Make magazine location ready for loading, unloading tool</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td>T number</td>
<td></td>
</tr>
<tr>
<td>WVar2:</td>
<td>INT</td>
<td>Location_number_from</td>
<td></td>
</tr>
<tr>
<td>WVar3:</td>
<td>INT</td>
<td>Magazine_number_from</td>
<td></td>
</tr>
<tr>
<td>WVar4:</td>
<td>INT</td>
<td>Location_number_to</td>
<td></td>
</tr>
<tr>
<td>WVar5:</td>
<td>INT</td>
<td>Magazine_number_to</td>
<td></td>
</tr>
</tbody>
</table>
13.17.4.27 PI service: TMPOSM

Function: Position magazine location or tool, depending on the parameter assignment

A magazine location, which has either been specified directly or via a tool located on it, is traversed to a specified position (e.g. in front of a load location) via the PI service.

The destination, e.g. the loading location, is specified in the parameters:

- WVar5 "Location number_ref"
- WVar6 "Magazine number_ref"

The magazine location to be positioned is specified in the following parameters, depending on the respective specification:

- WVar1 "T number" of the tool
  The following parameters are irrelevant here:
  - Addr1 "Tool name" = ""
  - WVar2 "Duplo number" = -1
  - WVar3 "Location number_From" = -1
  - WVar4 "Magazine number_From" = -1

- Addr1 "Tool name", WVar2 "Duplo number"
  The following parameters are irrelevant here:
  - WVar1 "T number of the tool" = -1
  - WVar3 "Location number_From" = -1
  - WVar4 "Magazine number_From" = -1

- WVar3 "Location number_From", WVar4 "Magazine number_From"
  The following parameters are irrelevant here:
  - Addr1 "Tool name" = ""
  - WVar1 "T number of the tool" = -1
  - WVar2 "Duplo number" = -1

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.TMPOSM</td>
<td>Position magazine location or tool</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>Addr1:</td>
<td>STRING</td>
<td>max. 32 characters</td>
<td>Tool name</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td>T number of the tool</td>
<td></td>
</tr>
<tr>
<td>WVar2:</td>
<td>INT</td>
<td>Duplo number</td>
<td></td>
</tr>
<tr>
<td>WVar3:</td>
<td>INT</td>
<td>Location_number_from</td>
<td></td>
</tr>
<tr>
<td>WVar4:</td>
<td>INT</td>
<td>Magazine_number_from</td>
<td></td>
</tr>
<tr>
<td>WVar5:</td>
<td>INT</td>
<td>Location_number_ref</td>
<td></td>
</tr>
<tr>
<td>WVar6:</td>
<td>INT</td>
<td>Magazine number_ref</td>
<td></td>
</tr>
</tbody>
</table>
13.17.4.28 PI service: TMPCIT

Function: Set increment value for workpiece counter

Incrementing the workpiece counter of the spindle tool

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.TMPCIT</td>
<td>Set increment value for workpiece counter</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>WVar1:</td>
<td>WORD</td>
<td>0, 1, 2 ... max.</td>
<td>Spindle number; corresponds to the type index in the location data with &quot;Spindle&quot; location type of the buffer magazine in channel.000 = main spindle</td>
</tr>
<tr>
<td>WVar2:</td>
<td>WORD</td>
<td>0 ... max.</td>
<td>Increment value; indicates the number of spindle revolutions after which the workpiece counter is incremented</td>
</tr>
</tbody>
</table>

13.17.4.29 PI service: TMRASS

Function: Resetting the tool status "active"

The PI service sets the status to "inactive" for all of the tools with a tool status "active" or "blocked".

The following events are sensible times for resetting the tool status:

- a negative edge of the NC/PLC interface signal "tool disable ineffective"
- End of program
- Channel reset.

The PI service is intended mainly for the PLC, since it is known here when the disabled tool is finally no longer to be used.

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.TMRASS</td>
<td>Reset active status</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TO area</td>
</tr>
</tbody>
</table>

13.17.4.30 PI service: TRESMO

Function: Reset monitoring values

This PI service resets the monitoring values of the designated edges of the designated tools to their setpoint (initial) values.

This is only performed for tools with active monitoring.

See also the RESETMON command.
Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;, TRESMO</td>
<td>Reset monitoring values</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TO area</td>
</tr>
<tr>
<td>WVar1:</td>
<td>WORD</td>
<td>- max ... max</td>
<td>Tool number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Process all tools</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&gt; 0: process specified tool</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt; 0: Process all sister tools</td>
</tr>
<tr>
<td>WVar2:</td>
<td>WORD</td>
<td>0, 1, 2 ... max.</td>
<td>D number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt; 0: Reset of the monitoring of the specified cutting edge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Reset of monitoring of all cutting edges</td>
</tr>
<tr>
<td>WVar3:</td>
<td>WORD</td>
<td>Monitoring mode that is to be reset (all combinations are possible):</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 0 = 1: Tool-life monitoring</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 1 = 1: Quantity monitoring</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 2 = 1: Wear monitoring</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 3 = 1: Sum-offset monitoring</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: All active tool-monitoring functions ($TC_TP9).</td>
</tr>
</tbody>
</table>

13.17.4.31 **PI service: TSEARC**

**Function:** Complex search using search screen form, depending on the parameter assignment

The PI service is used to search for tools with specific properties within a search range in one or more magazines, beginning with a specific location, up to a specific location.

**Note**

**Active tool management**

The PI service is only available if tool management is activated.

**Specification options**

- Search direction
- Search for next tool with the specified property
- Search for all tools with the specified property

**Result**

As a result, a list with the internal T-numbers of the tools that are found is created.

**Logic operations**

For filtering properties, only one AND link is available as a linking option. An OR link must be achieved by the user via several calls of the PI service and subsequent evaluation of the individual results.
Parameterization of the tool properties

The properties of the searched for tools are set in the BTSS block TF (parameterization, return parameters from TMGETT, TSEARC) via the following variables:

- "parMask<X>" (parameterization input screen)
- "parData<X>" (comparison values)

With <X> = TAD, TAO, TAS, TD

Result list

After completion of the PI service without errors, the search result is located in the BTSS block TF (parameterization, return parameter from TMGETT, TSEARC) in the following variables:

- "resultCuttingEdgeNrUsed" (D numbers of cutting edges used since last quantity count)
- "resultNrOfCutEdgesUsed" (number of cutting edges since last quantity count)
- "resultNrOfTools" (number of found tools)
- "resultToolNr" (T numbers of found tools)
- "resultToolNrUsed" (T numbers of cutting edges used since last quantity count)

If no tool was found, the number of found tools is zero ("resultNrOfTools" == 0)

Search range specifications

<table>
<thead>
<tr>
<th>From mag. number wVar1</th>
<th>From location number wVar2</th>
<th>To mag. number wVar3</th>
<th>To location number wVar4</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#M&lt;a&gt;</td>
<td>#L&lt;b&gt;</td>
<td>#M&lt;x&gt;</td>
<td>#L&lt;y&gt;</td>
<td>From: Magazine #M&lt;a&gt;, location #L&lt;b&gt; to: Magazine #M&lt;x&gt;, location #L&lt;y&gt;</td>
</tr>
<tr>
<td>#M&lt;a&gt;</td>
<td>-1</td>
<td>#M&lt;a&gt;</td>
<td>-1</td>
<td>From: Magazine #M&lt;a&gt;, first location to: Magazine #M&lt;a&gt;, last location</td>
</tr>
<tr>
<td>#M&lt;a&gt;</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>From: Magazine #M&lt;a&gt;, first location to: Last magazine, last location</td>
</tr>
<tr>
<td>#M&lt;a&gt;</td>
<td>#L&lt;b&gt;</td>
<td>-1</td>
<td>-1</td>
<td>From: Magazine #M&lt;a&gt;, location #L&lt;b&gt; to: Last magazine, last location</td>
</tr>
<tr>
<td>#M&lt;a&gt;</td>
<td>#L&lt;b&gt;</td>
<td>#M&lt;a&gt;</td>
<td>-1</td>
<td>From: Magazine #M&lt;a&gt;, location #L&lt;b&gt; to: Magazine #M&lt;a&gt;, last location</td>
</tr>
<tr>
<td>#M&lt;a&gt;</td>
<td>#L&lt;b&gt;</td>
<td>#M&lt;x&gt;</td>
<td>-1</td>
<td>From: Magazine #M&lt;a&gt;, location #L&lt;b&gt; to: Magazine #M&lt;x&gt;, last location</td>
</tr>
<tr>
<td>#M&lt;a&gt;</td>
<td>-1</td>
<td>#M&lt;x&gt;</td>
<td>#L&lt;y&gt;</td>
<td>From: Magazine #M&lt;a&gt;, first location to: Magazine #M&lt;x&gt;, location #L&lt;y&gt;</td>
</tr>
<tr>
<td>#M&lt;a&gt;</td>
<td>-1</td>
<td>#M&lt;x&gt;</td>
<td>-1</td>
<td>From: Magazine #M&lt;a&gt;, first location to: Magazine #M&lt;x&gt;, last location</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>From: First magazine, first location to: Last magazine, last location</td>
</tr>
</tbody>
</table>
Symmetrical search

To ensure that a symmetrical search can be carried out relative to a tool location, the following prerequisites must be met:

- The search range must encompass only one magazine: "WVar1" (from: magazine number) == "WVar3" (to: magazine number)
- Specification of a reference location, i.e. a tool location around which the symmetrical search will take place: "WVar5" (number of the reference magazine) and "WVar6" (number of the reference location)
- For the reference location, a multiple assignment to the magazine to be searched must have been configured in the TPM block.
- "WVar7" (search direction) = 3

The reference location is a buffer storage location, i.e. a tool location from the magazine buffer storage, e.g. swapping station, grippers, etc., or a loading station/location, i.e. a tool location from the internal loading magazine. The symmetrical search takes place with regard to the tool location, which is upstream of the reference location.

If the tool location is upstream of the reference location outside of the search range, the PI service will respond as if no suitable location had been found.

Note

Before and after this PI service, the MMCSEM PI service must be called up with the associated parameter WVar1 for this PI service. See Chapter "PI service: MMCSEM (Page 950)".

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.TSEARCH</td>
<td>Complex search using search screen forms</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td></td>
<td>From: Magazine number</td>
</tr>
<tr>
<td>WVar2:</td>
<td>INT</td>
<td></td>
<td>From: Location number</td>
</tr>
<tr>
<td>WVar3:</td>
<td>INT</td>
<td></td>
<td>to: Magazine number</td>
</tr>
<tr>
<td>WVar4:</td>
<td>INT</td>
<td></td>
<td>to: Location number</td>
</tr>
<tr>
<td>WVar5:</td>
<td>INT</td>
<td></td>
<td>Number of the reference magazine (only relevant for symmetrical search: Search direction ==3)</td>
</tr>
<tr>
<td>WVar6:</td>
<td>INT</td>
<td></td>
<td>Number of the reference location (only relevant for symmetrical search: Search direction ==3)</td>
</tr>
</tbody>
</table>
### Signal Description

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
</table>
| WVar7:  | INT  | 1, 2, 3     | Search direction:  
1: Forwards from the first location of the search domain  
2: Backwards from the last location of the search domain  
3: Symmetrical to the real magazine location, which is located upstream of the location specified under "WVar5" (number of the reference magazine) and "WVar6" (number of the reference location) |
| WVar8:  | INT  | 0, 1, 2, 3  | Search type  
Search for the property in the search range:  
0: all tools, cutting edge-specific  
1: first tool, cutting edge-specific  
2: all tools, via all cutting edges  
3: first tool, via all cutting edges |

### PI service: TMCRMT

#### Function: Create multitool

The PI service is used to create a new multitool with a defined identifier, an optionally specifiable multitool number, the number of tool locations, and the type of distance coding.

#### Note

Before and after this PI service, the PI service MMCSEM with parameter "WVar1" =1 (TMCRTO) must be called. See Chapter "PI service: MMCSEM (Page 950)".

### Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.TMCRMT</td>
<td>Create multitool</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>Addr1:</td>
<td>STRING</td>
<td></td>
<td>Multitool designator (max. 32 characters)</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td>0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
| WVar2:   | INT      | -1, 1, 2 ... 32000 | Multitool number  
-1: Automatic assignment of the multitool number by NC  
1, 2, 3 ... 32000: Multitool number,  
**note:** The multitool number must be unique across the whole system |
| WVar3:   | INT      | 2, 3, 4 ... MAX | Number of tool locations  
MAX = Parameterized number in MD17504  
$MN_MAX_TOOLS_PER_MULTITOOL |
| WVar4:   | INT      | 1, 2, 3     | Type of distance coding |
13.17.4.33 PI service: TMDLMT

Function: Delete multitool

The PI service is used to delete a multitool in all of the data blocks in which it is saved. Tools equipped in multitool are then no longer equipped and no longer loaded, but they are still defined.

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.TMDLMT</td>
<td>Delete multitool</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>Addr1:</td>
<td>STRING</td>
<td>Multitool designator (max. 32 characters)</td>
<td></td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>WVar2:</td>
<td>INT</td>
<td>-1, 1, 2 ... 32000</td>
<td>Multitool number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1: Delete the multitool with the name specified in &quot;Addr1&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1, 2, 3 ... 32000: Delete the multitool with the specified multitool number;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>note:</strong> Parameter &quot;Addr1&quot; is not evaluated</td>
<td></td>
</tr>
<tr>
<td>WVar3:</td>
<td>INT</td>
<td>0, 1</td>
<td>Tools contained in the multitool:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Do not delete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Delete</td>
<td></td>
</tr>
</tbody>
</table>

13.17.4.34 PI service: POSMT

Function: Position multitool

The PI service is used for positioning a multitool at the programmed location or alternatively at the programmed tool, which is located in one of the locations of the multitool. The tool itself can either be specified using its T number or with its name plus duplo number. A multitool can only be positioned if it is at a toolholder location and if no tool offset with regard to this toolholder location is active.

Position specification

The position specification can be alternatively assigned via:

<table>
<thead>
<tr>
<th>No .</th>
<th>Addr1</th>
<th>WVar1</th>
<th>WVar2</th>
<th>WVar3</th>
<th>WVar4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Empty string</td>
<td>Number of the toolholder</td>
<td>Tool number</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Tool name</td>
<td>Number of the toolholder</td>
<td>-1</td>
<td>Duplo number</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Empty string</td>
<td>Number of the toolholder</td>
<td>-1</td>
<td>-1</td>
<td>Multitool location number</td>
</tr>
</tbody>
</table>
### Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIService:</td>
<td>ANY</td>
<td>&quot;PI&quot;.POSMT</td>
<td>Position multitool</td>
</tr>
<tr>
<td>Unit:</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>Addr1:</td>
<td>STRING</td>
<td></td>
<td>Tool name of the tool to be positioned in the multitool (max. 32 characters)</td>
</tr>
<tr>
<td>WVar1:</td>
<td>INT</td>
<td>1, 2, 3 ... 999</td>
<td>Number of the toolholder</td>
</tr>
</tbody>
</table>
| WVar2:     | INT         | -1, 1, 2, 3 ... 32000 | Tool number (T number) of the tool to be positioned in the multitool  
1, 2, 3 ... 32000: Tool number  
-1: The tool number is irrelevant, "Addr1" and "WVar3" must be programmed |
| WVar3:     | INT         | -1, 1, 2, 3 ... 32000 | Duplo number of the tool to be positioned in the multitool  
1, 2, 3 ... 32000: Duplo number  
-1: The duplo number is irrelevant, "WVar2" must be programmed |
| WVar4:     | INT         | 1, 2, 3 ... 999   | Multitool location number of the toolholder location to which the system should be positioned |

### 13.17.4.35 PI service: FDPLMT

**Function: Search/check an empty location within the multitool**

The PI service is used to search for a free tool location in a multitool to accommodate the specified tool or for checking whether the specified tool location in the multitool for accommodating the specified tool is free. The tool can be specified using its T number, or using its identifier and its duplo number.

**Note**

Before and after this PI service, the PI service MMCSEM with parameter "WVar1" = x (FDPLMT) must be called. See Chapter "PI service: MMCSEM (Page 950)".

**Position specification**

The tool to be positioned in the multitool can be alternatively assigned via:

- "Addr1" = <Empty string>, "WVar1" = <Tool number> and "WVar2" = <Duplo number>
- "Addr1" = <Tool name>, "WVar1" = -1 and "WVar2" = <Duplo number>
- "Addr1" = <Empty string>, "WVar1" = <Tool number> and "WVar2" = -1

The position specification can be alternatively assigned via:
### Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlService</td>
<td>ANY</td>
<td>&quot;PI&quot;.FDPLMT</td>
<td>Search/check an empty tool location within a multitool</td>
</tr>
<tr>
<td>Unit</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>TOA</td>
</tr>
<tr>
<td>Addr1</td>
<td>STRING</td>
<td></td>
<td>Tool name of the tool to be positioned in the multitool (max. 32 characters)</td>
</tr>
</tbody>
</table>
| WVar1      | INT   | -1, 1, 2, 3 ... 32000   | Tool number (T number) of the tool to be positioned in the multitool  
  -1: The tool number is irrelevant, "Addr1" and "WVar2" must be programmed  
  1, 2, 3 ... 32000: Tool number |
| WVar2      | INT   | -1, 1, 2, 3 ... 32000   | Duplo number of the tool to be positioned in the multitool  
  1, 2, 3 ... 32000: Duplo number  
  -1: The duplo number is irrelevant, "WVar1" must be programmed |
| WVar3      | INT   | -1, 1, 2, 3 ... 32000   | Number of the multitool  
  1, 2, 3 ... 32000: Number of the multitool  
  -1: Search of all multitools for an empty location or check of all multitools to see whether the specified tool location is free in one of them for accommodating the tool. |
| WVar4      | INT   | -1, 1, 2, 3 ... 999     | Multitool location number of the tool location to which the system should be positioned  
  1, 2, 3, ... 999: Multitool location number  
  -1: Search for any empty tool location within the multitool |

### 13.17.5 FB5: GETGUD - read GUD variable

#### Function

The function block FB5 "GETGUD" is used for reading global user data (GUD) in the NC or channel area.

In order to reference the NC variables, they are first selected with the "NC VAR selector" tool and generated as STL source in a data block. A name must then be assigned to this data block in the S7 symbol table. When calling FB2, the variable addresses are transferred in the following form: Parameter "Addr1" to "Addr8" = 
"<DB name>".<S7 name>"

#### Request for reading NC variables

Call of FB5 with positive edge change, parameter "Req" = 0 → 1

Parameter "Addr": The pointer to the name of the GUD variables, symbolically with "<Data block>".<Variable name>
Parameters "Area", "Unit", "Index1", and "Index2": Additional information for addressing the variables.

When the parameter "CnvToken" is activated, the user receives a token (variable pointer) for the GUD variable to be read. Using this, the GUD variables can then be read or written via FB2 and FB3 with parameter "Addr1" ... "Addr8" = "<Token>". Addressing by means of a token is mandatory for writing GUD variables. For GUD variable arrays, the parameter "Line1" ... "Line8" = "<array index>" must also be written.

To read a GUD variable of the DOUBLE type without adapting the format, an ANY pointer of the REAL2 type must be specified in the target area. E.g. P#M100.0 REAL 2. The value of the GUD variables of the DOUBLE type is then adopted in the PLC data area as a 64-bit floating point number.

**Completion of the read request**

Read request successfully completed: Parameter "Done" == 1.

Read request completed with error:"Parameter "Error" == 1, error cause in parameter "State"

**Prerequisites**

- Release of the NC/PLC communication by OB100, FB1 parameter "NCKomm" = 1
- For the data block DB120 (data interface), the S7 Symbol Editor must be used to assign a symbol (default: NCVAR) in the S7 symbol table of the S7 project. Via this symbol, the NC variable is then specified in the FB2 parameters "Addr<x>", e.g. "ADDR1": = "NCVAR".<NC-Variable>"

**General conditions**

- FB5 has multi-instance capability.
- Every call of FB5 must be assigned a separate instance DB from the user area.
- Reading of a GUD variable (FB5 call with Req = 1) is only permitted in the cyclic part of the PLC basic program (OB1). If the job is not started (FB5 call with Req = 0), the parameters can also be written in the start-up part of the PLC basic program (OB100). The job can then be executed using the already written parameters in the cyclic part of the PLC basic program (OB1) by calling FB5 with Req = 1.
- Only capital letters may be used for the names of GUD variables.
- Reading of a GUD variable generally extends over several PLC cycles.
- When channel-specific variables are read, only variables from exactly one channel may be addressed via "Addr1" to "Addr8" if FB2 is called.
- When drive-specific variables are read, only variables from exactly one SERVO drive object may be addressed via "Addr1" to "Addr8" if FB2 is called. The SERVO drive object must be assigned to a machine axis of the NC. The line index corresponds to the logical drive number.
- In a read job, only variables from the same area, channel, or drive object can be read.
Note

Error case
When reading variables from different channels or drive objects, or simultaneously from a channel and a drive object, the following feedback message is output:

- "Error" == TRUE
- "State" == W#16#02

Note

To read a variable of the DOUBLE type from NC without adapting the format, an ANY pointer of the REAL2 type must be specified in the target area (e.g.: P#M100.0 REAL 2). If the basic program recognizes REAL 2 as the target type when reading a variable of the DOUBLE type, the data is applied to the PLC data area as a 64-bit floating-point number.

Declaration of the function

```plaintext
FUNCTION_BLOCK FB5  //Server name
    KNOW_HOW_PROTECT
    VERSION : 3.0
VAR_INPUT
    Req :      BOOL;
    Addr:     ANY;  // Variable name
    Area     BYTE ;  //Area: NCK = 0, channel = 2
    Unit :     BYTE ;
    Index1:   INT;   // Array index 1
    Index2:   INT;   // Array index 2
    CnvtToken: BOOL;  // Conversion into 10-byte token
    VarToken  ANY;   // Struct with 10 bytes for the variable token
END_VAR

VAR_OUTPUT
    Error :      BOOL;
    Done :      BOOL;
    State :     WORD;
END_VAR

VAR_IN_OUT
    RD:         ANY;
END_VAR

BEGIN
END_FUNCTION_BLOCK
```
### Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req:</td>
<td>I</td>
<td>BOOL</td>
<td>Job start with positive signal edge</td>
</tr>
<tr>
<td>Addr:</td>
<td>I</td>
<td>ANY</td>
<td>Variable name in a variable of the type STRING</td>
</tr>
<tr>
<td>Area:</td>
<td>I</td>
<td>BYTE</td>
<td>Range:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0, 2</td>
<td>0: NC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: Channel</td>
</tr>
<tr>
<td>Unit:</td>
<td>I</td>
<td>BYTE</td>
<td>Area == NC: Unit:=1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1, 2, 3, ... 10</td>
<td>Area == Channel: Channel number</td>
</tr>
<tr>
<td>Index1:</td>
<td>I</td>
<td>INT</td>
<td>Array index 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0, 1, 2, ... &lt;max. array index&gt;</td>
<td>For variables without array index: Index1 = 0</td>
</tr>
<tr>
<td>Index2:</td>
<td>I</td>
<td>INT</td>
<td>Array index 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0, 1, 2, ... &lt;max. array index&gt;</td>
<td>For variables without 2nd array index: Index2 = 0</td>
</tr>
<tr>
<td>CnvtToken:</td>
<td>I</td>
<td>BOOL</td>
<td>Activate generation of a 10 byte variable token</td>
</tr>
<tr>
<td>VarToken:</td>
<td>I</td>
<td>ANY</td>
<td>Address of a 10byte token (see example)</td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>BOOL</td>
<td>Negative acknowledgement of job or execution of job impossible</td>
</tr>
<tr>
<td>Done:</td>
<td>O</td>
<td>BOOL</td>
<td>Job successfully executed</td>
</tr>
<tr>
<td>State:</td>
<td>O</td>
<td>WORD</td>
<td>See paragraph &quot;Error identifiers&quot;</td>
</tr>
<tr>
<td>RD:</td>
<td>I/O</td>
<td>ANY</td>
<td>Target area for read data</td>
</tr>
</tbody>
</table>

#### Error identifiers

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD H 1)</td>
<td>WORD L</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1 Access error</td>
<td>Incorrect compilation of Var. in a job</td>
</tr>
<tr>
<td>0</td>
<td>2 Error in job</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3 Negative acknowledgement, job not executable</td>
<td>Internal error, try: NC RESET</td>
</tr>
<tr>
<td>0</td>
<td>4 Data areas or data types do not tally</td>
<td>Check data to be read in RD</td>
</tr>
<tr>
<td>1</td>
<td>4 Insufficient local user memory available</td>
<td>read variable is longer than specified in RD</td>
</tr>
<tr>
<td>0</td>
<td>6 FIFO full</td>
<td>Job must be repeated, since queue is full</td>
</tr>
<tr>
<td>0</td>
<td>7 Option not set</td>
<td>BP parameter &quot;NCKomm&quot; is not set</td>
</tr>
<tr>
<td>0</td>
<td>8 Incorrect target area (SD)</td>
<td>RD may not be local data</td>
</tr>
<tr>
<td>0</td>
<td>9 Transmission occupied</td>
<td>Job must be repeated</td>
</tr>
</tbody>
</table>
### Configuration steps

To be able to read a GUD variable, its name must be stored in a string variable. The data block with this string variable must be defined in the symbol table so that the "Addr" parameter can be assigned symbolically for FB GETGUD. A structure variable can be defined optionally in any data area of the PLC to receive the variable pointer (see specification in following example).

### Pulse diagram

1. User: Set request, Req = 0 → 1
2. FB4: PI service successfully completed, Done = 1  
   User: Reset request, IF Done == 1 THEN Req = 0
3. User: IF Done == 1 THEN reset request: 1 → 0
4. FB4: Reset job confirmation, Done = 0
5. User: IF Done == 0 AND Error == 0 THEN reset request Req = 1 → 0 **not permissible**
6. FB4: PI service completed with errors, Error = 1  
   User: Reset request, IF Done == 1 OR Error == 1 THEN Req = 0, possible further error handling

### Call example 1

Read a GUD variable from channel 1:
- Name "GUDVAR1"
- Type: INTEGER
Conversion to a 10 byte variable pointer. See table in "Assignment of the data types" in Chapter "FB2: GET - read NC variable (Page 922)"

**Reading the GUD variables: FB5 with instance DB111**

```plaintext
// Data block for GUD variable
DATA_BLOCK DB_GUDVAR    // Assignment in symbol table

STRUCT
  GUDVar1 : STRING[32] := 'GUDVAR1';    // Name is defined by user
  GUDVar1Token :
    STRUCT
      SYNTAX_ID : BYTE ;
      area_and_unit : BYTE ;
      column : WORD;
      line : WORD;
      block type : BYTE ;
      NO. OF LINES : BYTE ;
      type : BYTE ;
      length : BYTE ;
    END_STRUCT;
  END_STRUCT;
END_STRUCT;

BEGIN
END_DATA_BLOCK

// Unassigned user DB, as instance for FB5
DATA_BLOCK DB111

FB5
BEGIN
END_DATA_BLOCK

// Unassigned user DB, as instance for FB3
DATE_BLOCK DB112

FB3
BEGIN
END_DATA_BLOCK

// Reading of a channel-specific GUD variable from channel 1, with conversion to a variable pointer
Function FC "VariablenCall" : VOID
  U    I 7.7;     // Unassigned machine control panel key
  S    M 100.0;   // Activate req.
  U    M 100.1;   // Done completed message
  R    M 100.0;   // Terminate job
```
U   I 7.6;    // Manual error acknowledgment
U   M 102.0;  // Error pending
R   M 100.0;  // Terminate job

CALL FB5, DB111(
    Req := M 100.0, // Starting edge for reading
    Area := B#16#2, // Channel variable
    Unit := B#16#1, // Channel 1
    Index1 := 0,   // No array index 1
    Index2 := 0,   // No array index 2
    CnvtToken := TRUE,  // Request: Conversion into 10-byte
token
    VarToken := "DB_GUDVAR".GUDVar1Token, // Address of the token
    Error := M 102.0,
    Done := M 100.1,
    State := MW 104
    RD := P#DB99.DBX0.0 DINT 1  // free memory area
);

Writing the GUD variables: FB3 with instance DB112
GUD variable token from FB5, parameter: "VarToken" for writing with FB3, parameter "Addr1"

CALL FB3, DB112(
    Req := M 200.0,
    NumVar := 1,   // one GUD variable
    Addr1 := "DB_GUDVAR".GUDVar1Token, // Token
    Error := M 102.0,
    Done := M 100.1,
    State := MW 104
    SD1 := P#DB99.DBX0.0 DINT 1
);

Call example 2
Read a GUD variable from channel 1:
• Name "GUD_STRING"
• Type: STRING[32]
Conversion to a 10 byte variable pointer.

Reading the GUD variables: FB5 with instance DB111

// Data block for GUD variable
DATA_BLOCK DB_GUDVAR  // Assignment in symbol table

STRUCT
    GUDVarS : STRING[32] := 'GUD_STRING';  // Name defined by user
    GUDVarSToken :
STRUCT
SYNTAX_ID : BYTE ;
area_and_unit : BYTE ;
column : WORD;
line : WORD;
block type : BYTE ;
NO. OF LINES : BYTE ;
type : BYTE ;
length : BYTE ;
END_STRUCT;

string_of_GUD : STRING[30];    // must at least be so long as
                                 // the definition of 'GUD_STRING'!
END_STRUCT;

BEGIN
END_DATA_BLOCK

// Unassigned user DB, as instance for FB5
DATA_BLOCK DB111

FB5
BEGIN
END_DATA_BLOCK

// Unassigned user DB, as instance for FB3
DATE_BLOCK DB112

FB3
BEGIN
END_DATA_BLOCK

// Reading of a channel-specific GUD variable from channel 1, with conversion
to a variable pointer
Function FC "VariablenCall" : VOID
U    I 7.7;    // Unassigned machine control panel key
S    M 100.0;  // Activate req.
U    M 100.1;  // Done completed message
R    M 100.0;  // Terminate job
U    I 7.6;    // Manual error acknowledgment
U    M 102.0;  // Error pending
R    M 100.0;  // Terminate job
CALL FB5, DB111{ 
    Req := M 100.0,  // Starting edge for reading
Addr := "DB_GUDVAR".GUDVarS,
Area := B#16#2, // Channel variable
Unit := B#16#1, // Channel 1
Index1 := 0, // No array index
Index2 := 0, // No array index
CnvtToken := TRUE, // Request: Conversion into 10-byte token
VarToken := "DB_GUDVAR".GUDVarSToken, // Address of the token
Error := M 102.0,
Done := M 100.1,
State := MW 104
RD := "DB_GUDVAR".string_of_GUD
);

Writing the GUD variables: FB3 with instance DB112
GUD variable token from FB5, parameter: "VarToken" for writing with FB3, parameter "Addr1"

CALL FB3, DB112(
    Req := M 200.0,
    NumVar := 1, // one GUD variable
    Addr1 := "DB_GUDVAR".GUDVarSToken, // Token
    Error := M 102.0,
    Done := M 100.1
    State := MW 104
    SD1 := "DB_GUDVAR".new_name
);

13.17.6 FB7: PI_SERV2 - request PI service

Function
With the exception of a higher number of "WVar" parameters ("WVar11" - "WVar16"), function block FB7 has the same functionality as function block FB4. It is recommended that function block FB7 be used instead of function block FB4.

For a detailed description, refer to Chapter "FB4: PI_SERV - request PI service (Page 937)".

Declaration of the function

FUNCTION_BLOCK FB7
    Var_INPUT
        Req: BOOL;
        PI_Service: ANY;
        Unit: INT;
        Addr1: ANY;
    );
13.17.7 FB9: MtoN - operator panel switchover

Function

The function block FB3 "PUT" is used for switching over control components (MCP/OP), which are connected via a bus system to one or more control assemblies (NCU).

The interface between the individual control panels and the NCU (PLC) is the M : N interface in data block DB19. FB9 uses the signals of these interfaces.

Apart from initialization, sign-of-life monitoring and error routines, the following basic functions are also performed by the block for control unit switchover:

<table>
<thead>
<tr>
<th>Tabulated overview of functions:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic function</strong></td>
</tr>
<tr>
<td>HMI queuing</td>
</tr>
<tr>
<td>HMI coming</td>
</tr>
<tr>
<td>HMI going</td>
</tr>
<tr>
<td>Forced break</td>
</tr>
</tbody>
</table>
### Tabulated overview of functions:

<table>
<thead>
<tr>
<th>Basic function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating focus changeover to server mode</td>
<td>Change operating focus from one NCU to the other</td>
</tr>
<tr>
<td>Active/passive operating mode:</td>
<td>Operator control and monitoring/monitoring only</td>
</tr>
<tr>
<td>MCP switchover</td>
<td>As an option, MCP can be switched over with the HMI</td>
</tr>
</tbody>
</table>

### Note

The block must be called by the user program. The user must provide an instance DB with any number for this purpose. The call is multi-instance-capable.

### Brief description of a few important functions

**Active/passive operating mode:**

An online HMI can operate in two different modes:

- **Active mode:** Operator can control and monitor
- **Passive mode:** Operator can monitor (HMI header only)

After switchover to an NCU, this initially requests active operating mode in the PLC of the online NCU. If two control units are linked online simultaneously to an NCU, one of the two is always in active mode and the other in passive mode. The operator can request active mode on the passive HMI at the press of a button.

**MCP switchover**

As an option, an MCP assigned to the HMI can be switched over at the same time. To achieve this, the MCP address must be entered in the "mstt_adress" parameter of the NETNAMES.INI configuration file on the HMI and "MCPEnable" must be set to TRUE. The MCP of the passive HMI is deactivated so that there is only ever one active MCP on an NCU at one time.

**Boot condition**

To prevent the previously selected MCP being reactivated when the NCU is restarted, input parameters `<MCP1BusAdr = 255>` (address of 1st MCP) and "MCP1STOP" = TRUE (deactivate 1st MCP) must be set when FB1 is called in OB100.
Approvals

When one MCP is switched over to another, any active feed or axis enables will be retained.

Note

Keys actuated at the moment of switchover remain operative until the new MCP is activated (by the HMI, which is subsequently activated). The override settings for feedrate and spindle also remain valid. To deactivate actuated keys, the input image of the machine control signals must be switched to nonactuated signal level on a falling edge of DB10.DBX104.0 (MCP 1 ready). The override settings should remain unchanged. Measures for deactivating keys must be implemented in the PLC user program (see example "Override Changeover").

The call is permitted only in cyclic program OB1.

Declaration of the function

```
FUNCTION_BLOCK FB9
VAR_INPUT
  Ackn: BOOL; // Acknowledge interrupts
  OPMixedMode: BOOL:= FALSE; // Mixed operation with non-M-to-N-enabled OP // deactivated
  ActivEnable: BOOL:=TRUE; // Not supported
  MCPEnable: BOOL:=TRUE; // Activate MCP switchover
END_VAR
VAR_OUTPUT
  Alarm1: BOOL; // Interrupt: Error in HMI bus address, bus type!
  Alarm2: BOOL; // Interrupt: No confirmation HMI 1 offline!
  Alarm3 : BOOL; // Interrupt: HMI 1 is not going offline!
  Alarm4 : BOOL; // Interrupt: No confirmation HMI 2 offline!
  Alarm5 : BOOL; // Interrupt: HMI 2 is not going offline!
  Alarm6 : BOOL; // Interrupt: Queueing HMI is not going online!
  Report : BOOL; // Message: Signoflife monitoring
  ErrorMMC : BOOL; // Error detection HMI
END_VAR
```

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ackn:</td>
<td>I</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Acknowledge interrupts</td>
</tr>
<tr>
<td>OPMixedMode:</td>
<td>I</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Mixed operation deactivated for OP without M:N capability</td>
</tr>
</tbody>
</table>
### Signal Descriptions

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivEnable</td>
<td>I</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Function is not supported. Operator panel switch over interlocking using MMCx_SHIFT_LOCK in DB19</td>
</tr>
<tr>
<td>MCPEnable</td>
<td>I</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Activate MCP switch over. 1: MCP is switched over with operator panel. 0: MCP is not switched over with operator panel. This can be used to permanently link an MCP. See also MMCx_MSTT_SHIFT_LOCK in DB19.</td>
</tr>
<tr>
<td>Alarm1</td>
<td>O</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Interrupt: Error in HMI bus address, bus type!</td>
</tr>
<tr>
<td>Alarm2</td>
<td>O</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Interrupt: No confirmation HMI 1 offline!</td>
</tr>
<tr>
<td>Alarm3</td>
<td>O</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Interrupt: HMI 1 is not going offline!</td>
</tr>
<tr>
<td>Alarm4</td>
<td>O</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Interrupt: No confirmation HMI 2 offline!</td>
</tr>
<tr>
<td>Alarm5</td>
<td>O</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Interrupt: HMI 2 is not going offline!</td>
</tr>
<tr>
<td>Alarm6</td>
<td>O</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Interrupt: Queuing HMI is not going online!</td>
</tr>
<tr>
<td>Report</td>
<td>O</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Message: Sign-of-life monitoring HMI</td>
</tr>
<tr>
<td>ErrorMMC</td>
<td>O</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Error detection HMI</td>
</tr>
</tbody>
</table>

### Example of calling FB9

```plaintext
CALL FB9, DB109(
  Ackn := Error_ack, // e.g., MCP RESET
  OPMixedMode := FALSE,
  ActivEnable := TRUE,
  MCPEnable := TRUE); // Enable for MCP switchover
```

### Note

Input parameter "MCPEnable" must be set to TRUE to enable the MCP switchover. The default value of these parameters is set in this way and need not be specially assigned when the function is called.

### Intermits, errors

The output parameters "Alarm1" to "Alarm6" and "Report" exist as information in the PLC and are output in the event of M:N errors visualized on the HMI by the appearance of alarms 410900 - 410906.

If execution of an HMI function has failed (and an appropriate error message cannot be displayed), status parameter "ErrorMMC" is set to 'logical 1' (e.g., booting error, when no connection is made).
Example of a call for FB (call in the OB100)

CALL "RUN_UP", "gp_par" ( M#CNPn := 1,
M#CP1In := P#I 0.0,
M#CP1Out := P#Q 0.0,
M#CP1StatSend := P#Q 8.0,
M#CP1StatRec := P#Q 12.0,
M#CP1BusAdr := 255,  // Address of 1st MCP
M#CP1Timeout := S5T#700MS,
M#CP1Cycl := S5T#200MS,
M#CP1Stop := TRUE,  // MCP switched off
N#CyclTimeout := S5T#200MS,
N#CRunupTimeout := S5T#50S);

Example: Override switchover

// Auxiliary flags used M100.0, M100.1, M100.2, M100.3
// Edge positive of MCP1Ready must check the override
// and measures for activation
// Initiate MCP block
// This example applies to the feedrate override;
// The interface and input bytes must be exchanged for spindle override.
U  DB10.D#BX104.0;  //MCP1Ready
EN  M 100.0;  // Edge memory bit 1
JCN smth1;
S  M 100.2;  // Set auxiliary memory bit 1
R  M 100.3;  // Reset auxiliary flag 2

// Save override
L  DB21.D#BB4;  //Feed override interface
T  EB28;  //Buffer storage (freely input
// or flag byte)

weil:
U  M 100.2;  //Switchover takes place
O  DB10.D#BX104.0;  //MCP1Ready
JCN smth2;
U  DB10.D#BX104.0;  //MCP1Ready
FP  M 100.1;  // Edge memory bit 2
JC  smth2;
U  M 100.2;  //Switchover takes place
R  M 100.2;  // Reset auxiliary flag 1
JC  smth2;
U  M 100.3;  //Comparison has taken place
SPB MCP;  //Call MCP program
// Route the stored override to the interface of the switched MCP
// until the override values match
L EB28;  //Buffer storage open
T DB21.DBB4;  //Route override interface
L EB3;  //Override input byte for feed
<>i;  //Match?
JC smth2;  //No, jump
S M 100.3;  //Yes, set auxiliary flag 2
// When override values match, call the MCP program again
MCP: CALL "MCP_IFM"(  //FC19
  BAGNo := B#16#1,
  ChanNo := B#16#1,
  SpindleIFNo := B#16#0,
  FeedHold := M 101.0,
  SpindleHold := M 101.1);
wei2: NOP 0;

13.17.8  FB10: Safety relay (SI relay)

Function

The SPL function block FB10 "Safety relay" for "Safety Integrated" is the PLC equivalent of
the NC function of the same name. The standard SPL "Safety relay" block is designed to
support the implementation of an emergency stop function with safe programmable logic.
However, it can also be used to implement other similar safety functions, e.g., control of a
protective door.

The function contains 3 input parameters ("In1", "In2", "In3"). On switchover of one of these
parameters to the value 0, the output "Out0" is deactivated without delay and outputs "Out1",
"Out2", and "Out3" are deactivated via the parameterized timer values (parameters
"TimeValue1", "TimeValue2", "TimeValue3"). The outputs are activated again without delay if
inputs "In1" to "In3" take on value 1 and a positive edge change is detected at one of the
acknowledgment inputs "Ack1", "Ack2".

Call bit memory

To bring the outputs to their basic setting (values = 0) after booting, the parameter "FirstRun"
must be configured as follows. The parameter "FirstRun" must be switched to the value TRUE
via a retentive data (memory bit, bit in data block) on the 1st run after control booting. This
data can be preset, e.g., in OB100. The parameter "FirstRun" is reset to FALSE when FB10
is executed for the first time. Separate data must be used for parameter "FirstRun" for each
call with its own instance.
General conditions

- FB10 has multi-instance capability
- After the start of the SPL program, FB10 is called once per SI relay in the cyclic part of the PLC basic program (OB1).
- Every call of FB10 must be assigned a separate instance DB from the user area.

Simplified block diagram in CSF

The figure below shows only one acknowledgment input "Ack1" and one delayed deactivation output "Out1". The circuit for "Ack2" and the other delayed outputs are identical. The parameter "FirstRun" is also missing in the function diagram. The mode of operation is described above.

Declaration of the function

FUNCTION_BLOCK FB10
VAR_INPUT
  In1 : BOOL;
  In2 : BOOL;
  In3 : BOOL;
  Ackn1 : BOOL;
  Ackn2 : BOOL;
Description of formal parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>In1:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Input 1</td>
</tr>
<tr>
<td>In2:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Input 2</td>
</tr>
<tr>
<td>In3:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Input 3</td>
</tr>
<tr>
<td>Ackn1:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Acknowledgment input 1</td>
</tr>
<tr>
<td>Ackn2:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Acknowledgment input 2</td>
</tr>
<tr>
<td>TimeValue1:</td>
<td>I</td>
<td>S5TIME</td>
<td></td>
<td>Time value 1 for OFF delay</td>
</tr>
<tr>
<td>TimeValue2:</td>
<td>I</td>
<td>S5TIME</td>
<td></td>
<td>Time value 2 for OFF delay</td>
</tr>
<tr>
<td>TimeValue3:</td>
<td>I</td>
<td>S5TIME</td>
<td></td>
<td>Time value 3 for OFF delay</td>
</tr>
<tr>
<td>Out0:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Output, instantaneous (no delay)</td>
</tr>
<tr>
<td>Out1:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Output, delayed by TimeValue1</td>
</tr>
<tr>
<td>Out2:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Output, delayed by TimeValue2</td>
</tr>
<tr>
<td>Out3:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Output, delayed by TimeValue3</td>
</tr>
<tr>
<td>FirstRun:</td>
<td>I/O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Call bit memory</td>
</tr>
</tbody>
</table>

13.17.9 FB11: Brake test

Function

The braking operation check should be used for all axes, which must be prevented from moving in an uncontrolled manner by a holding brake. "Vertical axes" are the main application.

The machine manufacturer can use the PLC user program to regularly close the brake at a suitable instant (e.g. every 8 hours) and have the drive exert torque/force in addition to the weight of the axis. In errorfree operation, the brake can produce the necessary braking torque/braking force. The axis will hardly move during this.
When an error occurs, the actual position value exits the parameterizable monitoring window. The position controller prevents sagging of the axis. The function test of the brake mechanical system is negatively acknowledged.

References
A detailed description of the parameterization of the NC and drive within the framework of the "Safety Integrated" function can be found in:
Function Manual for Safety Integrated

Starts the brake test
The brake test must always be started when the axis is at a standstill. For the entire duration of the brake test, the enable signals of the parameterized axis must be set to enable (e.g. controller inhibit, feed enable). Furthermore, the signal at the axis/spindle DB31, ... .DBX28.7 (PLC-controlled axis) is to be set to status 1 by the user program for the complete duration of the test.

Before activating the NC/PLC interfaces DB31, ... .DBX28.7 (PLC-controlled axis), the axis is to be switched as "neutral axis", e.g. DB31, ... .DBX8.0 - 8.3 (assign NC axis to channel) is to be set to channel 0, as well as DB31, ... .DBX8.4 (activation signal when changing this byte) is to be set.

The return message:
• about the current status can be queried in DB31, ... DBB68.
• the Nc via the signal DB31, ... .DBX63.1 (PLC controls axis) is to be awaited before the block is started. The direction in which the drive must produce its torque/force is specified by the PLC in the form of a "traversing motion" (e.g., via FC18).

The axis must be able to reach the destination of this movement without risk of collision if the brake is unable to produce the necessary torque/force.

Note
Note on FC18
If FC18 is called over the further course of the user program for the same axis, the calls must be mutually interlocked. For example, this can be achieved via a common call of this function with an interlocked common data interface for the FC18 parameters. A second option is to call the FC18 repeatedly, in which case the inactive FC18 will not be processed by the program. A multiple-use interlock must be provided.

Structure of a brake test

<table>
<thead>
<tr>
<th>Step</th>
<th>Expected feedback</th>
<th>Monitoring time value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Start brake test</td>
<td>DBX71.0 = 1</td>
<td>TV_BTactiv</td>
</tr>
<tr>
<td>2 Close brake</td>
<td>Bclosed = 1</td>
<td>TV_Bclose</td>
</tr>
<tr>
<td>3 Output traversing command</td>
<td>DBX64.6 OR DBX64.7</td>
<td>TV_FeedCommand</td>
</tr>
<tr>
<td>4 Issue test travel command</td>
<td>DBX62.5 = 1</td>
<td>TV_FXReached</td>
</tr>
<tr>
<td>5 Wait for the holding time</td>
<td>DBX62.5 = 1</td>
<td>TV_FXShold</td>
</tr>
<tr>
<td>6 Deselect brake test / open brake</td>
<td>DBX71.0 = 0</td>
<td>TV_BTactiv</td>
</tr>
<tr>
<td>7 Output test ok</td>
<td>***</td>
<td>***</td>
</tr>
</tbody>
</table>
General conditions

- FB2 has multi-instance capability.
- Every call of FB11 must be assigned a separate instance DB from the user area.

Declaration of the function

Function_BLOCK FB11

VAR_INPUT
  Start            BOOL;
  Ackn :           BOOL;
  Bclosed :        BOOL;
  Axis :           INT;
  TimerNo :        TIMER ;
  TV_BTactiv :     S5TIME ;
  TV_Bclose :      S5TIM;
  TV_FeedCommand : S5TIME ;
  TV_FXSreached :  S5TIME ;
  TV_FXShold :     S5TIME ;
END_VAR

VAR_OUTPUT
  CloseBrake :     BOOL;
  MoveAxis :       BOOL;
  Done :           BOOL;
  Error :          BOOL;
  State :          BYTE ;
END_VAR

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>I</td>
<td>BOOL</td>
<td>Starts the brake test</td>
</tr>
<tr>
<td>Ackn</td>
<td>I</td>
<td>BOOL</td>
<td>Acknowledge fault</td>
</tr>
<tr>
<td>Bclosed</td>
<td>I</td>
<td>BOOL</td>
<td>Checkback input whether Close Brake is activated (singlechannel - PLC)</td>
</tr>
<tr>
<td>Axis</td>
<td>I</td>
<td>INT</td>
<td>Axis number of axis to be tested</td>
</tr>
<tr>
<td>TimerNo</td>
<td>I</td>
<td>TIMER</td>
<td>Timer from user program</td>
</tr>
<tr>
<td>TV_BTactiv</td>
<td>I</td>
<td>S5TIME</td>
<td>Monitoring time value → brake test active, check of axis signal DBX71.0</td>
</tr>
<tr>
<td>TV_Bclose</td>
<td>I</td>
<td>S5TIME</td>
<td>Monitoring time value → close brake Check of input signal Bclosed after output CloseBrake has been set.</td>
</tr>
<tr>
<td>TV_FeedCommand</td>
<td>I</td>
<td>S5TIME</td>
<td>Monitoring time value → Travel command given Check travel command after MoveAxis has been set</td>
</tr>
<tr>
<td>TV_FXSreached</td>
<td>I</td>
<td>S5TIME</td>
<td>Monitoring time value → fixed stop reached</td>
</tr>
</tbody>
</table>
Fault IDs

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>1</td>
<td>Start conditions not fulfilled, e.g. the axis is not in closed-loop control / brake closed / axis inhibited</td>
</tr>
<tr>
<td>2</td>
<td>No NC checkback in &quot;Brake test active&quot; signal on selection of brake test</td>
</tr>
<tr>
<td>3</td>
<td>No &quot;Brake applied&quot; checkback by input signal Bclosed</td>
</tr>
<tr>
<td>4</td>
<td>No travel command output (e.g., axis motion has not been started)</td>
</tr>
<tr>
<td>5</td>
<td>Fixed end stop will not be reached → axis RESET was initiated.</td>
</tr>
<tr>
<td>6</td>
<td>Traversing inhibit/Approach too slow → fixed stop cannot be reached. TV FXSreached monitoring timeout</td>
</tr>
<tr>
<td>7</td>
<td>Brake is not holding at all (the end position is reached)/approach speed is too high</td>
</tr>
<tr>
<td>8</td>
<td>Brake opens during the holding time</td>
</tr>
<tr>
<td>9</td>
<td>Error when deselecting the brake test</td>
</tr>
<tr>
<td>10</td>
<td>Internal error</td>
</tr>
<tr>
<td>11</td>
<td>&quot;PLC-controlled axis&quot; signal not enabled in the user program</td>
</tr>
</tbody>
</table>

Example of calling FB11:

```plaintext
UN  M  111.1;  //Request to close brake, Z axis of FB
=  O  85.0;  //Brake control, Z axis
AUF Axis3";  //Brake test, Z axis
O  I  73.0;  //Brake test trigger, Z axis
O  M  110.7;  //Brake test running
FP  M  110.0;
UN  M  111.4;  //Error has occurred
S  M  110.7;  //Brake test running
S  M  110.6;  //Next step
JCN  m001
L  DBB  68;
AW  W#16#F;
T  MB  115;  //flag channel state
L  B#16#10
T  DBB  8;  //Request neutral axis
```
m001:  U   DBX  68.6;  //Checkback signal, axis is neutral
U   M  110.6;
FP  M  110.1;
R   M  110.6;
S   M  110.5;  //Next step
S   DBX 28.7;  //Request PLC-monitored axis

U   DBX  63.1;  //Checkback signal, axis monitored by PLC
U   M  110.5;
FP  M  110.2;
R   M  110.5;
S   M  111.0;  //Start brake test for FB

CALL FB11, DB211 (//brake test block
Start  :=M  111.0,  //Start brake test
Ackn   :=I  3.7,   //Acknowledge error with RESET key
Bclosed :=I  54.0, //Return message close brakes
          //controlled
Axis    :=3,      //Axis number of axis to be tested
          //Z axis
TimerNo :=T  110,  //Timer number
TV_BTactiv :=S5T#200MS,  //Monitoring time value:
          //Brake test active DBX71.0
TV_Bclose :=S5T#1S,   //Monitoring time value:
          //Brake closed
TV_FeedCommand :=S5T#1S,  //Monitoring time value:
          //Traversing command output
TV_FXSreache :=S5T#1S,  //Monitoring time value:
          //Fixed stop reached
TV_FXShold :=S5T#2S,   //Monitoring time value:
          //Brake test time
CloseBrake :=M  111.1,  //Request to close brake
MoveAxis :=M  111.2,  //Initiate //Request traversing motion //
Done   :=M  111.3,   //Test successfully completed
Error  :=M  111.4,   //Error has occurred
State  := MB 112);  //Error status

AUF "Axis3";  //Brake test, Z axis

U   M  111.2;  //Moveaxis
FP  M  111.5;  //FC18 Start
S   M  111.7;  //Start FC18
O   M  111.3;  //Test successfully completed
O M 111.4; //Error has occurred
FP M 110.3;
R DBX 28.7; //Request, PLC-monitored axis

UN DBX 63.1; //Checkback signal, axis monitored by PLC
U M 111.0; //Start brake test for FB
U M 110.7; //Brake test running
FP M 110.4;
R M 111.0; //Start brake test for FB
R M 110.7; //Brake test running

//optional begin
JCN m002:
L MB 115; //old channel status
OW W#16#10;
T DBB 8; //Request channel axis
m002: NOP 0;
//optional end

CALL "SpinCtrl" //(Traverse Z axis
Start :=M 111.2, //Start traversing motion
Stop := FALSE,
Funct := B#16#5, //Mode: Axis mode
Mode := B#16#1, //Procedure: Incremental
AxisNo := 3, //Axis number of axis to be traversed
//axis Z-axis
Pos := -5.000000e+000, //Traversing distance: Minus 5 mm
FRate := 1.000000e+003, //Feedrate: 1000 mm/min
InPos :=M 113.0, //Position reached
Error :=M 113.1, //Error has occurred
State := MB 114); //Error status

AUF "Axis3"; //Brake test, Z axis
U M 113.0; //Position reached
O M 113.1; //Error has occurred
FP M 113.2;
R M 111.7; //Start FC18
13.17.10  FB29: Signal recorder and data trigger diagnostics

Function

Signal recorder
FB29 "Diagnostics" allows various diagnostic routines to be performed on the PLC user program. A diagnostic routine logs signal states and signal changes. In this diagnostic routine, function number 1 is assigned to the "Func" parameter. Up to 8 signals of the parameters "Signal_1" to "Signal_8" are recorded in a ring buffer each time one of the signals changes. The current information of parameters "Var1" as BYTE value, and "Var2" and "Var3" as INTEGER values are also stored in the ring buffer. The number of past OB1 cycles is also stored in the buffer as additional information. This information enables the graphical evaluation of signals and values in OB1 cycle grid.

Call rule
First call of FB29 in OB1 cycle: Parameter "NewCycle" = 1
All other calls of FB29 in the same OB1 cycle: Parameter "NewCycle" = 0

Ring buffer
The ring buffer, which must be defined by the user, must have an ARRAY structure specified as in the source code. The array can have any number of elements. A size of 250 elements is recommended. The "ClearBuf" parameter is used to clear the ring buffer and set the "BufAddr" pointer to the start. The instance DB related to the FB29 is a DB from the user area and is to be transferred to the FB Diagnostics with the parameter "BufDB".

Data trigger
The data trigger function is intended to allow triggering on specific values (or bits) at any permissible memory cell. The cell to be triggered is "rounded" with a bit mask ("AndMask" parameter) before the "TestVal" parameter is compared in the diagnostic block.

Note
The source code for the function is available in the source container of the basic-program library under the name "Diagnose.awl". The instance DB and the ring buffer DB are also defined in this source block. The function call is also described in the function. The DB numbers and the call must be modified.

Declaration of the function

```
FUNCTION_BLOCK FB29
VAR_INPUT
  Func                      : INT;
  Signal_1                  : BOOL;
  Signal_2                  : BOOL;
  Signal_3                  : BOOL;
```
Structure of the ring buffer

TITLE = //Ring buffer DB for FB29
VERSION : 1.0

STRUCT
  Field: ARRAY [0 .. 249] OF
  //can be any size of this struct
  STRUCT
    Cycle : INT; //Delta cycle to last storage in buffer
    Signal_1 : BOOL; //Signal names same as FB29
    Signal_2 : BOOL;
    Signal_3 : BOOL;
    Signal_4 : BOOL;
    Signal_5 : BOOL;
    Signal_6 : BOOL;
    Signal_7 : BOOL;
    Signal_8 : BOOL;
    Var1 : BYTE ;
    Var2 : WORD;
    Var3 : WORD;
Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Func:</td>
<td>I</td>
<td>INT</td>
<td>0, 1, 2</td>
<td>Function</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Switch off</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Signal recorder</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: Data trigger</td>
</tr>
<tr>
<td>Parameters for function 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal_1 ... Signal_8:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Bit signals checked for change</td>
</tr>
<tr>
<td>NewCycle:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>See paragraph &quot;Function&quot; above</td>
</tr>
<tr>
<td>Var1 :</td>
<td>I</td>
<td>BYTE</td>
<td></td>
<td>Additional value</td>
</tr>
<tr>
<td>Var2 :</td>
<td>I</td>
<td>INT</td>
<td></td>
<td>Additional value</td>
</tr>
<tr>
<td>Var3 :</td>
<td>I</td>
<td>INT</td>
<td></td>
<td>Additional value</td>
</tr>
<tr>
<td>BufDB:</td>
<td>I</td>
<td>INT</td>
<td></td>
<td>Ring buffer DB no.</td>
</tr>
<tr>
<td>ClearBuf:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Delete ring buffer DB and reset pointer BuAddr</td>
</tr>
<tr>
<td>BufAddr:</td>
<td>I/O</td>
<td>INT</td>
<td></td>
<td>Target area for read data</td>
</tr>
<tr>
<td>Parameters for function 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DataAddr:</td>
<td>I</td>
<td>POINTER</td>
<td></td>
<td>Pointer to word to be tested</td>
</tr>
<tr>
<td>TestVal:</td>
<td>I</td>
<td>WORD</td>
<td></td>
<td>Comparison value</td>
</tr>
<tr>
<td>AndMask:</td>
<td>I</td>
<td>WORD</td>
<td></td>
<td>See paragraph &quot;Function&quot; above</td>
</tr>
<tr>
<td>TestIsTrue:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Result of comparison</td>
</tr>
</tbody>
</table>

Configuration steps

- Select function of diagnostics block.
- Define suitable data for the recording as signal recorder or data triggering.
- Find a suitable point or points in the user program for calling the diagnostics FB.
- Create a data block for the ring buffer, see call example.
- Call the diagnostics FB with parameters in the user program.

In function 1, it is advisable to clear the ring buffer with the "ClearBuf" parameter. When the recording phase with function 1 is completed, read out the ring buffer DB in STEP7 with the function "opening the data block in the data view". The content of the ring buffer DB can now be analyzed.
Call example

FUNCTION FC99: VOID
TITLE =
VERSION : 0.0
BEGIN
NETWORK
TITLE = NETWORK
CALL FB29, DB80(
Func := 1,
Signal_1 :=M 100.0,
Signal_2 :=M 100.1,
Signal_3 :=M 100.2,
Signal_4 :=M 100.3,
Signal_5 :=M 10.4,
Signal_6 :=M 100.5,
Signal_7 :=M 100.6,
Signal_8 :=M 100.7,
NewCycle := TRUE,
Var1 := MB 100,
BufDB := 81,
ClearBuf :=M 50.0);
END_FUNCTION

13.17.11 FC2 : GP_HP - basic program, cyclic section

Function

The NC/PLC interface is processed by the basic program in cyclic mode (OB1). To keep the
runtime to a minimum, only the control and status signals are cyclically transferred. The transfer
of auxiliary and G-functions is only processed upon request from the NC.

Furthermore, the data for handwheel selection, modes, and other operating signals are
transferred from the operator panel (HMI) to the NC/PLC interface so that the modes support
the selection from the MCP or HMI as required.

The transfer of HMI signals to the NC/PLC interface can be deactivated by setting the value
of the parameter "MMCToIF" to "FALSE" in FB1 (DB7).

Handwheel selection signals
Requirement: FB1, parameter "HWheelMMC == "TRUE"

The handwheel selection signals from the HMI are decoded and activated in the respective
machine or geometry axis of the respective handwheel.
Declaration

FUNCTION FC2: VOID
// no parameters

Call example

As far as the time is concerned, the basic program must be executed before the user program. It is, therefore, called first in OB1.

The following example contains the standard declarations for OB1 and the calls for the basic program (FC2), the transfer of the MCP signals (FC19), and the acquisition of error and operating messages (FC10).

ORGANIZATION_BLOCK OB1
VAR_TEMP
  OB1_EV_CLASS : BYTE ;
  OB1_SCAN_1 : BYTE ;
  OB1_PRIORITY : BYTE ;
  OB1_OB_NUMBR : BYTE ;
  OB1_RESERVED_1 : BYTE ;
  OB1_RESERVED_2 : BYTE ;
  OB1_PREV_CYCLE : INT ;
  OB1_MIN_CYCLE : INT ;
  OB1_MAX_CYCLE : INT ;
  OB1_DATE_TIME : DATE_AND_TIME ;
END_VAR
BEGIN
CALL FC2; //Call basic program as first FC
//INSERT USER PROGRAM HERE
CALL FC19( //MCP signals to interface
  BAGNo := B#16#1, // Mode group no. 1
  ChanNo := B#16#1, // Channel no. 1
  SpindleIFNo := B#16#4, //Spindle interface number = 4
                  // (Number of the associated machine axis)
  FeedHold := m22.0, //Feed stop signal
               //modal
  SpindleHold := db2.dbx151.0); //Spindle stop modal
                         //in message DB
CALL FC10( // Error and operating messages
  ToUserIF := TRUE, //Signals transferred from DB2
               //to interface
  Ackn := I6.1); //Acknowledgment of error messages
                  //via I6.1
END_ORGANIZATION_BLOCK
13.17.12 FC3: GP_PRAL - basic program, interruptdriven section

**Function**

Block-synchronized transfers from the NCK to the PLC (auxiliary and G functions) are carried out in the interrupt-driven part of the basic program. **Auxiliary functions** are subdivided into normal and highspeed auxiliary functions.

The highspeed functions of an NC block are buffered and the transfer acknowledged to the NC. These are transferred to the application interface at the start of the next OB1 cycle.

Highspeed auxiliary functions programmed immediately one after the other, are not lost for the user program. This is ensured by a mechanism in the basic program.

Normal auxiliary functions are acknowledged to the NC only after one completed cycle duration. This allows the application to issue a read disable to the NC.

The **G Functions** are evaluated immediately and passed to the application interface.

**NC process interrupts**

If the interrupt is triggered by the NC (possible in each IPO cycle), a bit in the local data of OB40 ("GP_IRFromNCK") is set by the basic program, when the FB1 parameter "UserIR" := TRUE"). This data is not set on other events (process interrupts through I/Os). This information makes it possible to branch into the associated interrupt routine in the user program in order to initiate the necessary action.

To be able to implement highspeed, job-driven processing of the user program for the machine, the following NC functions are available in the interrupt processing routine (OB40 program section) for the PLC user program:

- **Selected auxiliary functions**
- **Tool-change function** for tool-management option
- **Position reached** for positioning axes, indexing axes and spindles with activation via PLC

The functions listed above can or must be evaluated by the user program in OB40 in order to initiate reactions on the machine. As an example, the revolver switching mechanism can be activated when a T command is programmed on a turning machine.

For further details on programming hardware interrupts (time delay, interruptibility, etc.) refer to the corresponding SIMATIC documentation.

**Auxiliary functions**

Generally, high-speed or acknowledging auxiliary functions are processed with or without interrupt control independently of any assignment.

Basic-program parameters in FB1 can be set to define which auxiliary functions (T, H, DL) must be processed solely on an interrupt-driven basis by the user program.

Functions which are not assigned via interrupts are only made available by the cyclic basic program as in earlier versions. The change signals of these functions are available in a PLC cycle.
Even if the selection for the auxiliary function groups (T, H, DL) is made using interrupt control, only one interrupt can be processed by the user program for the selected functions.

A bit is set channel-specifically in the local data "GP_AuxFunction" for the user program (if "GP_AuxFunction[1]" is set, then an auxiliary function is available for the 1st channel).

In the related channel-DB the change signal and the function value are available for the user. The request signal of this interrupt-driven function is reset to zero in the cyclic basic program section after the execution of at least one full OB1 cycle (max. approx. two OB1 cycles).

**Tool change**

With the tool-management option, the tool-change command for revolver and the tool change in the spindle is supported by an interrupt. The local data bit "GP_TM" in OB40 is set for this purpose. The PLC user program can thus check the tool management DB (DB72 or DB73) for the tool change function and initiate the tool change operation.

**Position reached**

In the bit structure, "GP_InPosition" of the local data of OB40 is specific to the machine axis (each bit corresponds to an axis/spindle, e.g. GP_InPosition[5] corresponds to axis 5).

If a function has been activated via FC18 (spindle control, positioning axis, indexing axis) for an axis or spindle, the associated "GP_InPosition" bit can be used to implement instantaneous evaluation of the "InPos" signal of the FCs listed above. This feature can be used, for example, to obtain immediate activation of clamps for an indexing axis.

**Declaration**

```plaintext
FUNCTION FC3: VOID

// no parameters
```

**Call example**

As far as the time is concerned, the basic program must be executed before other interrupt-driven user programs. It is, therefore, called first in OB40.

The following example contains the standard declarations for OB40 and the call for the basic program.

```plaintext
ORGANIZATION_BLOCK OB40
VAR_TEMP
  OB40_EV_CLASS : BYTE  ;
  OB40_STRT_INF : BYTE  ;
  OB40_PRIORITY : BYTE  ;
  OB40_OB_NUMBR : BYTE  ;
  OB40_RESERVED_1 : BYTE  ;
  OB40_MDL_ID : BYTE  ;
  OB40_MDL_ADDR : INT;
  OB40_POINT_ADDR : DWORD;
```

---

P3: Basic PLC program for SINUMERIK 840D sl

13.17 Block descriptions


13.17.13 FC5: GP_DIAG - basic program, diagnostic alarm and module failure

Function

The block FC5 "GP_DIAG" is used to record assembly disruptions and failures.

A PLC stop can be triggered via the parameter "PlcStop". The PLC stop is only triggered for incoming events. The MCPs connected to the parameterized PROFIBUS (DP1) at FB1 are excluded from this.

Declaration

FUNCTION FC5: VOID
VAR_INPUT
   PlcStop: BOOL:= TRUE;
END_VAR

Call example

The basic program should be run through after processing of the user programs. This is recommended because a PLC stop can be triggered by the FC5.

The example contains the standard declaration for OB82 and OB86 and the call of the FC5.

ORGANIZATION_BLOCK OB82
VAR_TEMP
   OB82_EV_CLASS : BYTE ;
   OB82_FLT_ID : BYTE ;
   OB82_PRIORITY : BYTE ;
OB82_OB_NUMBR : BYTE ;
OB82_RESERVED_1 : BYTE ;
OB82_IO_FLAG : BYTE ;
OB82_MDL_ADDR : INT ;
OB82_MDL_DEFECT : BOOL ;
OB82_INT_FAULT : BOOL ;
OB82_EXT_FAULT : BOOL ;
OB82_PNT_INFO : BOOL ;
OB82_EXT_VOLTAGE : BOOL ;
OB82_FLD_CONNCTR : BOOL ;
OB82_NO_CONFIG : BOOL ;
OB82_CONFIG_ERR : BOOL ;
OB82_MDL_TYPE : BYTE ;
OB82_SUB_NDL_ERR : BOOL ;
OB82_COMM_FAULT : BOOL ;
OB82_MDL_STOP : BOOL ;
OB82_WTCH_DOG_FLT : BOOL ;
OB82_INT_PS_FLT : BOOL ;
OB82_PRIM_BATT_FLT : BOOL ;
OB82_BCKUP_BATT_FLT : BOOL ;
OB82_RESERVED_2 : BOOL ;
OB82_RACK_FLT : BOOL ;
OB82_PROC_FLT : BOOL ;
OB82EPROM_FLT : BOOL ;
OB82_RAM_FLT : BOOL ;
OB82_ADU_FLT : BOOL ;
OB82_FUSE_FLT : BOOL ;
OB82_HW_INTR_FLT : BOOL ;
OB82_RESERVED_3 : BOOL ;
OB82_DATE_TIME : DATE_AND_TIME ;
END_VAR
BEGIN
CALL FC5
   (PlcStop := FALSE) ;
END_ORGANIZATION_BLOCK

ORGANIZATION_BLOCK OB86
VAR_TEMP
   OB86_EV_CLASS : BYTE ;
   OB86_FLT_ID : BYTE ;
   OB86_PRIORITY : BYTE ;
   OB86_OB_NUMBR : BYTE ;
   OB86_RESERVED_1 : BYTE ;
   OB86_RESERVED_2 : BYTE ;
   OB86_MDL_ADDR : WORD ;
OB86_RACKS_FLTD : ARRAY [0 .. 31]OF BOOL;
OB86_DATE_TIME : DATE_AND_TIME;
END_VAR
BEGIN
    CALL FC5
        (PlcStop := TRUE) ;
END_ORGANIZATION_BLOCK

13.17.14 FC6: TM_TRANS2 - transfer block for tool management and multitool

Function
The block FC6 "TM_TRANS2" is used for position changes of the tools, status changes, and multitool.
The FC6 block has the same functionality as the FC8 block, plus the multitool functionality.
The description of FC6 only contains the multitool functionality.
The functionality of FC8 is described in "FC8: TM_TRANS - transfer block for tool management (Page 1000)".

Declaration of the function

FUNCTION FC6: VOID
VAR_INPUT
    Start: BOOL;
    TaskIdent: BYTE ;
    TaskIdentNo: BYTE ;
    NewToolMag: INT;
    NewToolLoc: INT;
    OldToolMag: INT;
    OldToolLoc: INT;
    Status: INT;
    MtoolPlaceNum: INT;
END_VAR
VAR_OUTPUT
    Ready: BOOL;
    Error: INT;
END_VAR
BEGIN
END_FUNCTION
Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>See block description FC8</td>
</tr>
<tr>
<td>TaskIdent:</td>
<td>I</td>
<td>BYTE</td>
<td></td>
<td>See block description FC8</td>
</tr>
<tr>
<td>TaskIdentNo:</td>
<td>I</td>
<td>BYTE</td>
<td></td>
<td>See block description FC8</td>
</tr>
<tr>
<td>NewToolMag:</td>
<td>I</td>
<td>INT</td>
<td></td>
<td>See block description FC8</td>
</tr>
<tr>
<td>NewToolLoc:</td>
<td>I</td>
<td>INT</td>
<td></td>
<td>See block description FC8</td>
</tr>
<tr>
<td>OldToolMag:</td>
<td>I</td>
<td>INT</td>
<td></td>
<td>See block description FC8</td>
</tr>
<tr>
<td>OldToolLoc:</td>
<td>I</td>
<td>INT</td>
<td></td>
<td>See block description FC8</td>
</tr>
<tr>
<td>Status:</td>
<td>I</td>
<td>INT</td>
<td></td>
<td>See block description FC8</td>
</tr>
<tr>
<td>MtoolPlaceNum:</td>
<td>I</td>
<td>INT</td>
<td></td>
<td>Multitool location No.</td>
</tr>
<tr>
<td>Ready:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>See block description FC8</td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>INT</td>
<td></td>
<td>See block description FC8</td>
</tr>
</tbody>
</table>

13.17.15 FC7: TM_REV - transfer block for tool change with revolver

Function

After a revolver has been changed, the user will call the block FC7 "FC TM_REV". The revolver number corresponding to interface number in DB73 must be specified in parameter "ChgdRevNo" for this purpose. As this block is called, the associated "Interface active" bit in data block DB73.DBW0 of FC7 is reset after parameter "Ready" == TRUE is returned.

Job executed correctly

If the job was executed correctly, then "Ready" == 1. The user must then set the parameter "Start" = 0 or no longer call FC7.

Job executed with errors

If the job was executed with errors, then parameter "Ready" == 0 and parameter "Error" == 1. The job must be repeated in the next PLC cycle. Since the parameter "Start" does not need a positive edge for a subsequent job, "Start" = 1 remains, because the job has not yet been completed. See "Call example" and "Pulse diagram" below.

General conditions

- Block FC7 may only be started with parameter "Start" = 1 if an activation signal for the associated interface (DB73.DBW0) for this transfer has been supplied by the tool management function.
- A cancellation of a transfer, e.g. by a channel reset, is not permitted.
- Parameter "Start" = 1, until parameter "Ready" == 1 or "Error" == 1
References

- For detailed information about tool management, refer to the Function Manual Tool Manager.
- PI services for tool management
  - FB4: Request PI service (Page 937)
  - FC8: TM_TRANS - transfer block for tool management (Page 1000)
  - FC22: TM_DIR - direction selection for tool management (Page 1044)

Manual revolver switching

If the revolver is rotated in manual operation, neither a tool change nor an offset selection is associated with this operation. The first step is the removal of the tool from the toolholder back to its location in the revolver. An asynchronous transfer must be performed with FC8 (alternative: FC6). The associated parameter settings are shown below:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaskIdent</td>
<td>4</td>
</tr>
<tr>
<td>TaskIdentNo</td>
<td>Channel no.</td>
</tr>
<tr>
<td>NewToolMag</td>
<td>Magazine no. of the revolver</td>
</tr>
<tr>
<td>NewToolLoc</td>
<td>Original location of the tool</td>
</tr>
<tr>
<td>OldToolMag</td>
<td>Magazine no. of the buffer storage (spindle) = 9998</td>
</tr>
<tr>
<td>OldToolLoc</td>
<td>Buffer storage no. of the spindle</td>
</tr>
<tr>
<td>Status</td>
<td>1</td>
</tr>
</tbody>
</table>

If the revolver is now turned to an arbitrary position at which a tool is located, this tool must be activated. This is done easiest by the new T programming in the part program. However, if this should take place, for example, at the end of revolver switching from the PLC user program, an ASUB must be started for this purpose. The current revolver position must be transferred to the ASUB. In this way, the tool at this location is determined in the ASUB and is selected (see Jobshop example in the toolbox).

Declaration of the function

```plaintext
FUNCTION FC7: VOID
//NAME :TM_REV
VAR_INPUT
    Start : BOOL;
    ChgdRevNo: BYTE ;
END_VAR
VAR_OUTPUT
    Ready: BOOL;
    Error : INT;
END_VAR
BEGIN
END_FUNCTION
```
Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1 = start transfer</td>
</tr>
<tr>
<td>ChgdRevNo:</td>
<td>I</td>
<td>BYTE</td>
<td>1, 2, 3, ...</td>
<td>Number of revolver interface</td>
</tr>
<tr>
<td>Ready:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1 = Transfer complete</td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>INT</td>
<td>0, 1, 2, 3</td>
<td>Error checkback</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error has occurred</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: No revolver present</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: Illegal revolver number in parameter &quot;ChgdRevNo&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3: Illegal job (&quot;interface active&quot; signal for selected revolver = &quot;FALSE&quot;)</td>
</tr>
</tbody>
</table>

Pulse diagram

1. User: Set request, Start = 0 → 1
2. FB4: PI service successfully completed, Ready = 1
   User: Reset request, IF Ready == 1 THEN Start = 0
3. User: IF Ready == 1 THEN reset request: 1 → 0
4. FB4: Reset job confirmation, Ready = 0
5. User: IF Ready == 0 AND Error == 0 THEN reset request Start = 1 → 0 not permissible
6. FB4: PI service completed with errors, Error = 1
   User: Reset request, IF Ready == 1 OR Error == 1 THEN Start = 0, possible further error handling

Call example

CALL // Tool management: Transfer block for revolver
FC7( 
   Start := m 20.5,  // Start := "1 " => initiate the transfer
   ChgdRevNo := DB61.DBB1, 
)
13.17.16 FC8: TM_TRANS - transfer block for tool management

Function

The user calls this block FC TM-TRANS when the position of the tool or the status of the transfer operation changes. The parameter "TaskIdent" specifies the transfer job for the block FC8 at the tool management interface:

- For loading/unloading positions
- For spindle change positions
- For revolver change positions as transfer identifier
- Asynchronous transfer
- Asynchronous transfer with location reservation

The interface number is indicated in parameter "TaskIdentNo".

Example for loading point 5:
Parameter "TaskIdent":= 1 and "TaskIdentNo":= 5.

Furthermore, the current tool positions and status data (list of "Status" parameter in the following text) are also transferred for this transfer function.

Note

FC8 informs the NCK of the current positions of the old tool.

The NCK knows where the old and the new tool have been located until the position change.

In the case of a transfer without a so-called "old tool" (e.g. on loading), the value 0 is assigned to parameters "OldToolMag", "OldToolLoc".
Block FC TM_TRANS may be started only with "Start" parameter = "TRUE" if an activation signal for the appropriate interface (DB71, DB72, DB73 in word 0) for this transfer has been supplied by the tool management function.

When this job is executed correctly, the output parameter "Ready" contains the value TRUE. The user must then set the "Start" parameter to FALSE or not call the block again.

If the "Ready" parameter = FALSE, the error code in the "Error" parameter must be interpreted (see Call example FC8 and pulse diagram).

If the error code = 0, then this job must be repeated in the next PLC cycle (e.g. "Start" remains set to "TRUE"). This means that the transfer operation has not yet been completed.

If the user assigns a value of less than 100 to the "Status" parameter, then the associated interface in data block DB71 or DB72 or DB73, word 0 is deactivated (process completed). The appropriate bit for the interface is set to 0 by FC8.

The "Start" parameter does not need a signal edge for a subsequent job. This means that new parameters can be assigned with "Start = TRUE" immediately when "Ready = TRUE" is received.

Asynchronous transfer

Asynchronous transfer ensures that changes in the position of a tool are automatically signaled from PLC to the tool management (e.g. power failure during an active command or independent changes in the position by the PLC), FC8 is called with "TaskIdent" = 4 or 5. This call does not require interface activation by tool management.

If parameter "TaskIdent" = 5, the tool management reserves the location in addition to changing the position. The location is only reserved if the tool has been transported from a real magazine to a buffer storage.

A relevant NC channel must be parameterized in the "TaskIdentNo" parameter.

The previous location of the tool is specified in the parameters "OldToolMag" and "OldToolLoc". The current location of the tool is specified in the parameters "NewToolMag", "NewToolLoc". "Status" = 1 must be specified.

With "Status" = 5, the specified tool remains at location "OldToolMag", "OldToolLoc". This location must be a buffer (e.g. spindle). The real magazine and location must be specified in the parameters "NewToolMag", "NewToolLoc"; the location is at the position of the buffer. This procedure must always be used if the tool management is to be informed of the position of a specific magazine location. This procedure is used for alignment in search strategies.

General conditions

- A cancellation of a transfer, e.g. by a channel reset, is not permitted.
- Parameter "Start" = 1, until parameter "Ready" == 1 or "Error" == 1
References

- For detailed information about tool management, refer to the Function Manual Tool Manager.
- PI services for tool management
  - FB4: Request PI service (Page 937)
  - FC7: TM_REV - transfer block for tool change with revolver (Page 997)
  - FC22: TM_DIR - direction selection for tool management (Page 1044)

Declaration of the function

FUNCTION FC8: VOID
//NAME :TM_TRANS
VAR_INPUT
  Start : BOOL;
  TaskIdent: BYTE ;
  TaskIdentNo: BYTE ;
  NewToolMag: INT;
  NewToolLoc: INT;
  OldToolMag: INT;
  OldToolLoc: INT;
  Status: INT;
END_VAR
VAR_OUTPUT
  Ready BOOL;
  Error : INT;
END_VAR
BEGIN
END_FUNCTION

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: Start transfer</td>
</tr>
<tr>
<td>TaskIdent:</td>
<td>I</td>
<td>BYTE</td>
<td>1, 2, 3, 4, 5</td>
<td>Interface or task identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Loading/unloading location</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: Spindle change position</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3: Revolver change position</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4: Asynchronous transfer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5: Asynchronous transfer with location reserva-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>tion</td>
</tr>
<tr>
<td>Signal</td>
<td>Type</td>
<td>Type</td>
<td>Value range</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
<td>-------</td>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TaskIdentNo:</td>
<td>I</td>
<td>BYTE</td>
<td>1, 2, 3 ... 10</td>
<td>Number of the associated interface or channel number. For the asynchronous transfer, the interface number can be specified in the upper nibble.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: 1. Interface, 2. Channel ⇒ B#16#12</td>
</tr>
<tr>
<td>NewToolMag:</td>
<td>I</td>
<td>INT</td>
<td>-1, 0, 1, 2 ...</td>
<td>Current magazine number of tool to be replaced</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1: Tool remains at its location</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NewToolLoc = any value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Only permissible if TaskIdent = 2</td>
</tr>
<tr>
<td>NewToolLoc:</td>
<td>I</td>
<td>INT</td>
<td>0, 1, 2 ... max.</td>
<td>Current location number of new tool</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>location no.</td>
<td></td>
</tr>
<tr>
<td>OldToolMag:</td>
<td>I</td>
<td>INT</td>
<td>-1, 0 ...</td>
<td>Current magazine number of tool to be replaced</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1: The tool remains at its location. &quot;OldToolLoc&quot; = &lt;any value&gt;. Only permissible if &quot;TaskIdent&quot; = 2</td>
</tr>
<tr>
<td>OldToolLoc:</td>
<td>I</td>
<td>INT</td>
<td>Max. location number</td>
<td>Current location number of tool to be replaced</td>
</tr>
<tr>
<td>Status:</td>
<td>I</td>
<td>INT</td>
<td>1, 2, 3 ... 7, 103,</td>
<td>Status information about transfer operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>104, 105</td>
<td></td>
</tr>
<tr>
<td>Ready:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: Transfer completed</td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>INT</td>
<td>0 ... 65535</td>
<td>Error checkback</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error has occurred</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Unknown &quot;TaskIdent&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: Unknown &quot;TaskIdentNo&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3: Impermissible job, (&quot;Interface active&quot; signal for selected revolver == 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Other value: The number corresponds to the error message of the tool management function in the NC caused by this transfer.</td>
</tr>
</tbody>
</table>
Pulse diagram

1. Activation of function by means of a positive edge
2. Positive acknowledgment: Tool management has been transferred
3. Reset function activation after receipt of acknowledgment
4. Signal change using FC
5. This signal chart is not permissible. The job must generally be terminated since the new tool positions must be conveyed to the tool management in the NCK.
6. Negative acknowledgement: Error occurred, error code in the output parameter Error
### Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1      | The WZV job has been completed.  
The parameters "NewToolMag", "NewToolLoc", "OldToolMag", "OldToolLoc" of the FC8 block should be parameterized to the actual positions of the tools involved. Except in the case of preparing change these are normally the specified target position of the tools of the associated WZV interface, see also "Explanations of the formal parameters".  
1. In the case of **loading/unloading/reloading**, the tool has arrived at the required target address. If the bit in the interface in DB 71.DBX (n+0).3 “position at loading point” is enabled, status 1 cannot be used for the function termination. Status 5 must be used for correct termination.  
2. In the case of **"Prepare change"**, the new tool is now available. The tool may, for example, be positioned in a buffer (gripper). In some cases, the target (magazine, location) of the old tool has been moved to the toolchange position after placement of the new tool in a buffer. However, the old tool still remains in the spindle. The preparations for a tool change are thus complete. After this acknowledgement, the "Change" command can be received. The positions in parameters "NewToolMag", "NewToolLoc", "OldToolMag" and "OldToolLoc" correspond to the current tool positions.  
3. In the case of **"Change"** (spindle or revolver), the tools addressed in the interface have now reached the required target addresses.  
The tool change operation is thus completed. |
| 2      | The "new" tool cannot be made available  
This status is only admissible in conjunction with the "Prepare Change" command. When this status is applied, the PLC must be prevented from making a change with the proposed tool. The proposed (new) tool is disabled by the tool management function in the NCK. A new command is then output by the tool management with a duplo tool. The positions in parameters "NewToolMag", "NewToolLoc", "OldToolMag", and "OldToolLoc" correspond to the original tool positions. |
| 3      | An error has occurred  
The tool positions must not have been changed. Any changes to the magazine positions that have taken place in the meantime must be notified beforehand, for example, with status = 105 via FC8 transfer block Only then will the tool positions be taken into account by the tool management function. |
| 4      | It would be better to position the "old" tool in the magazine position specified in parameters "OldToolMag" and "OldToolLoc"  
This status is permissible only in conjunction with preparation for tool change (change into spindle). After this status has been transferred to the tool management in the NCK, the tool management tries to consider the specified magazine position in the next command. But this is done only when this position is free. Parameters "NewToolMag" and "NewToolLoc" are not taken into account. |
<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td><strong>The operation is complete</strong>&lt;br&gt;The &quot;new&quot; tool is in the position specified in parameters &quot;NewToolMag&quot;, &quot;NewToolLoc&quot;. In this case, the specified tool is not really in this position, but is still in the same magazine location. However, this magazine location has been moved to the position set in the parameters (e.g. tool change position). This status may be used only for revolvers, chain-type magazines and disk magazines. The status enables the tool management function to adjust the current position of a magazine and to improve the search strategy for subsequent commands. This status is permissible only in conjunction with loading, unloading, and reloading operations and with preparations for a tool change. The &quot;OldToolMag&quot; and &quot;OldToolLoc&quot; parameters must be parameterized with the data of a buffer.</td>
</tr>
<tr>
<td></td>
<td>- Loading, reloading:&lt;br&gt;On loading or reloading, a location for the tool is already reserved in the NCK. The machine operator must then insert the tool at the target location. Notice: The location reservation is canceled when the control system is switched on again.</td>
</tr>
<tr>
<td></td>
<td>- Tool-change preparation:&lt;br&gt;Tool motions still to be executed are not carried out until after the tool has been changed.</td>
</tr>
<tr>
<td></td>
<td>- Positioning to load point:&lt;br&gt;If the bit in the interface in DB 71.DBX (n+0).3 &quot;position at loading point&quot; is enabled, then only status 5 be used for the function termination (not status 1).</td>
</tr>
<tr>
<td>6</td>
<td><strong>The WZV job has been completed</strong>&lt;br&gt;This status has the same function as status 1, but, in addition, a reservation of the source location is carried out. This status is only permitted when reloading. The command is ended and the source location of the tool is reserved if the target location is in a buffer magazine.</td>
</tr>
<tr>
<td>7</td>
<td><strong>Initiate repetition of the command &quot;Prepare Tool&quot;</strong>&lt;br&gt;This status is only admissible in conjunction with the &quot;Change tool&quot; command. This status is intended for use when the &quot;new&quot; tool has changed its position (e.g. via an asynchronous command of the &quot;new&quot; tool). After &quot;Ready = 1&quot; has been provided by FC8, the &quot;Prepare Change&quot; command is repeated automatically with the same tool. For the automatic repetition, a new tool search is carried out. The positions in parameters &quot;NewToolMag&quot;, &quot;NewToolLoc&quot;, &quot;OldToolMag&quot;, and &quot;OldToolLoc&quot; correspond to the original tool positions.</td>
</tr>
<tr>
<td>103</td>
<td><strong>The &quot;new&quot; tool can be inserted</strong>&lt;br&gt;This status is permitted only in the tool change preparation, when the PLC may reject the new tool (e.g. in case of MD20310 SMC_TOOL_MANAGEMENT_MASK, bit 4=1 for the possibility, request changed parameter from PLC once again). The tool positions have remained unchanged. This status is thus necessary, when the processing is to be continued in the NCK without an unnecessary stop.</td>
</tr>
<tr>
<td>104</td>
<td><strong>The &quot;new&quot; tool is in the position specified in parameters &quot;NewToolMag&quot;, &quot;NewToolLoc&quot;</strong>&lt;br&gt;This status is only permissible if the tool is still in the magazine in the same location. The &quot;old&quot; tool is in the position (buffer) specified in parameters &quot;OldToolMag&quot;, &quot;OldToolLoc&quot;. In this case, however, the new tool is not really in this position, but is still in the same magazine location. However, this magazine location has been moved to the position set in the parameters (e.g. tool change position). This status may be used only in conjunction with revolvers, chain-type magazines and disk magazines for the &quot;Tool change preparation&quot; phase. Status enables the tool management to adjust the current position of a magazine and to improve the search strategy for subsequent commands.</td>
</tr>
<tr>
<td>105</td>
<td><strong>The specified buffer location has been reached by all tools involved</strong>&lt;br&gt;Standard case if the operation has not yet been completed. The tools are in the specified tool positions (parameters &quot;NewToolMag&quot;, &quot;NewToolLoc&quot;, &quot;OldToolMag&quot;, &quot;OldToolLoc&quot;).</td>
</tr>
</tbody>
</table>
Status definition
A general rule for the acknowledgement status is that the status information 1 to 7 leads to the termination of the command. If FC8 receives one of the statuses, the "Interface active bit" of the interface specified in FC8 is reset to "0" (see also interface lists DB 71 to DB 73), thus completing the operation. The response is different in the case of status information 103 to 105. When the FC8 receives one of these items of status information, the "Interface active bit" of this interface remains at "1". Further processing is required by the user program in the PLC (e.g. continuation of magazine positioning). This item of status information is generally used to transfer changes in position of one or both tools while the operation is still in progress.

Call example

CALL FC8 ( //Tool-management transfer block
Start := m 20.5, // Start := "1" => initiate the transfer
TaskIdent := DB61.DBB0,
TaskIdentNo := DB61.DBB1,
NewToolMag := DB61.DBW2, // Current position of new tool
NewToolLoc := DB61.DBW4,
OldToolMag := DB61.DBW6, // Current position of old tool
OldToolLoc := DB61.DBW8,
Status := DB61.DBW10, // Status
Ready := m 20.6,
Error := DB61.DBW12);
u m 20.6; // Poll ready
r m 20.5; // Reset start
spb m001; // Jump if everything OK
l DB61.dbwl2; // Error information
ow w#16#0; // Evaluate error
JC error; // Jump to troubleshooting
m001: // Normal branch

error: //Troubleshooting
r m 20.5: // Reset start

13.17.17 FC9: ASUB - start of asynchronous subroutines

Function
The block FC9 "ASUP" can be used to trigger any functions in the NC. Before an ASUB can be started from the PLC, it must have been selected and parameterized by an NC program or by FB4 (PI service ASUB). In this case, the channel and the interrupt numbers must match the parameters in FC9.
Once prepared in this way, it can be started at any time from the PLC. The NC program running on the channel in question is interrupted by the asynchronous subprogram.

Only one ASUB can be started in the same channel at a time. If several ASUBs are started in one PLC cycle, the ASUBs are started in this order in the NC.

Parameter "Start" = 0 must be set by the user if the ASUB has been terminated ("Done" == 1) or an error has occurred ("Error" == 1).

For processing jobs, each FC9 needs its own parameter "Ref" from the global user area. This parameter is for internal use only and must not be changed by the user. The parameter "Ref" is initialized with the value 0 in the first OB1 cycle and, for this reason, every FC9 must be called absolutely. Alternatively, the user can initialize parameter "Ref" with a value of 0 during startup. This option makes conditional calls possible. A conditional call requires parameter "Start" = 1 during activation of FC9 until a negative edge change has occurred at parameter "Done" (1 → 0).

**General conditions**

- The function block FB4 must be terminated before the block FC9 is started.
- The block FC9 cannot be started if DB10, DBX56.1 == 1 (emergency stop).
- Block FC9 must not be started if channel reset is active in the channel in which the ASUB is to be started.

**Declaration of the function**

```plaintext
FUNCTION FC9: VOID
//NAME :ASUP
VAR_INPUT
    Start : BOOL;
    ChanNo: INT;
    IntNo: INT;
END_VAR
VAR_OUTPUT
    Active: BOOL;
    Done : BOOL;
    Error : BOOL;
    StartErr: BOOL;
END_VAR
VAR_IN_OUT
    Ref: WORD;
END_VAR
```
Description of formal parameters

The table below lists all formal parameters of the ASUB function.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Job start with positive signal edge</td>
</tr>
<tr>
<td>ChanNo:</td>
<td>I</td>
<td>INT</td>
<td>1, 2, 3 ... 10</td>
<td>Channel number</td>
</tr>
<tr>
<td>IntNo:</td>
<td>I</td>
<td>INT</td>
<td>1, 2, 3 ... 8</td>
<td>Interrupt number</td>
</tr>
<tr>
<td>Active:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: active</td>
</tr>
<tr>
<td>Done:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: ASUB completed</td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: Interrupt switched off</td>
</tr>
<tr>
<td>StartErr:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: Interrupt number not assigned or deleted</td>
</tr>
<tr>
<td>Ref:</td>
<td>I/O</td>
<td>WORD</td>
<td>Global variable (MW, DBW,..)</td>
<td>1 word per FC9 (for internal use)</td>
</tr>
</tbody>
</table>

Pulse diagram

1. Activation of function
2. ASUB is active
3. Positive acknowledgment: ASUB completed
4. Reset function activation after receipt of acknowledgment
5. Signal change using FC
6. Not permitted If function activation is reset prior to receipt of acknowledgement, the output signals are not updated without the operational sequence of the activated function being affected.
7. Negative acknowledgement: An error occurred
Call example

CALL FC9(  //Start an asynchronous subprogram  
    //in channel 1 interrupt number 1
    Start := I 45.7,  
    ChanNo := 1,  
    IntNo := 1,  
    Active := M 204.0,  
    Done := M204.1,  
    Error := M 204.4,  
    StartErr := M 204.5,  
    Ref := MW 200);

13.17.18 FC10: AL_MSG - error and operating messages

Function

With block FC10 "AL_MSG", the signals entered in DB2 are evaluated and displayed as incoming or outgoing error messages and operating messages on the user interface. The incoming signals (positive edge) are displayed immediately in the case of both error and operational messages. Outgoing signals (negative edge) are deleted immediately only for operating messages. In the event of error messages, the messages that are no longer pending are only deleted with the parameter "Quit", i.e. errors remain displayed on the user interface until they have been acknowledged by the user even if the signals are no longer pending.

The "ToUserIF" parameter can be used to transfer the group signals for the feed, read and NC start disabling signals and feed stop signal to the existing axis, spindle and channel interfaces. The group signals are transferred to the user interface directly from the status information in DB2 irrespective of an interrupt acknowledgment.

1. If parameter "ToUserIF" = 0, there is no transfer of the signals to the user interface. In this case, the user must take measures in his PLC program to ensure that these signals are influenced in the interface.

2. If parameter "ToUserIF" = 1, all signals listed above are sent to the user interface as a group signal in each case. The user PLC program can, therefore, influence these signals only via DB2 in conjunction with a message or interrupt output. The appropriate information is overwritten in the user interface.

Alternatively to the response described in paragraph 2, the disable and hold signals can be influenced without a message being output by influencing the interface signals with a disable or stop signal state after FC10 is called.

The following program illustrates this method:

CALL FC10(  
    ToUserIF := TRUE,  
    Ackn := I 6.1);
FB1 parameter ExtendAlMsg

With FB1 parameter "ExtendAlMsg" = 1, the new DB2 structure becomes effective (see "Interface PLC/HMI (Page 842)"). Upon activation, bit arrays are available for the disable and stop signals which do not generate any alarms or messages. As a result, the above measures do not need to be implemented. The desired functionality is automatically obtained by a simple setting/resetting signals in DB2.

The error and operating messages in data block DB2 must be provided in a user-specific way.

Note

In DB2, a "1" signal must be present for several OB1 cycles to ensure that a message can also be displayed on the HMI. There is an upper limit for the number of interrupts and messages that can be pending at the same time. This upper limit is dependent on the PLC CPU. On PLC 317-2DP, the upper limit for messages pending simultaneously is 60.

References

List Manual for NC variables and interface signals, Chapter "PLC user interface" > "PLC alarms/messages"
Call example

CALL FC10( // Error and operating messages
ToUserIF := TRUE, // Signals from DB2 are transferred to interface.
Ackn := I6.1); // Acknowledgment of error message via input I6.1

13.17.19 FC12: AUXFU - call interface for user with auxiliary functions

Function

The block FC12 "AUXFU" is called on an event-driven basis in the basic program if the channel transferred in the input parameter contains new auxiliary functions. The PLC user can extend FC AUXFU with program instructions for processing his auxiliary function to avoid cyclic polling of the channel DBs. This mechanism permits auxiliary functions to be processed on a job-driven basis. FC AUXFU is supplied as a compiled empty block in the basic program. In this case, the basic program supplies parameter "Chan" with the channel number. The PLC user knows which channel has new auxiliary functions available. The new auxiliary functions can be determined by the auxiliary-function change signals in the channel concerned.

Declaration of the function

FUNCTION FC12: VOID // Event control of auxiliary functions
VAR_INPUT
  Chan: BYTE ;
END_VAR
BEGIN
BE;
END_FUNCTION

Explanation of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chan:</td>
<td>I</td>
<td>BYTE</td>
<td>0, 1, 2 ... 9</td>
<td>Index of the channel = channel number -1</td>
</tr>
</tbody>
</table>

Example

FUNCTION FC12: VOID // Event control of auxiliary functions
VAR_INPUT
  Chan: BYTE ; // Parameter is supplied by basic program
END_VAR
VAR_TEMP
  ChanDB: INT;
END_VAR
BEGIN
L Chan;    // Channel index
+ 21;     // Channel DB offset
T ChanDB; // Save channel DB no.
TO DB[ChanDB]; // Channel DB is opened indirectly
// Auxiliary-function change signals are now scanned, etc.
BE;
END_FUNCTION

13.17.20 FC13: BHGDisp - display control for handheld unit

Function
This block FC13 "BHGDisp" handles the display control for the handheld unit (HHU or HT 2). The information that is to appear on the display must be saved to a string variable. The pointer to the string is specified in the parameter "ChrArray". To do this, a fixed text assignment of 32 characters (HHU) or 64 characters (HT 2) is needed when the data block for this string is created.

16 characters are sent to the HHU per job. The assignment of the characters in the "ChrArray" for the respective line is unambiguous. For line 1 the characters 1 to 16 and for the line 2 the characters 17 to 32 of the string data ChrArray are transferred. In addition, for HT 2 line 3 with characters 33 to 48 is displayed and line 4 with characters 49 to 64. A job takes several OB1 cycles.

Display
Block FC13 checks whether the necessary minimum length of the "ChrArray" exists for operating the handheld unit. If fewer characters exist in the string variables than should be displayed, the line is filled with blank spaces. If several variables are to be entered in the string in one or more PLC cycles without a display output, the display output can be suppressed by parameter "Row" = 0. The transfer of the characters to the rows takes several OB1 cycles. If several rows are to be updated "simultaneously" (parameter "Row" > 1), the rows are updated successively with 16 characters per row.

Variable portions
Variable portions within the string can be inserted using the optional number converter functionality with the parameter "Convert" = 1. The variable to be displayed is referenced via the parameter "Addr". The format of the variables is described in the parameter "DataType". The number of bytes of the variable is linked to the format description. The address justified
to the right within the string is specified by parameter "StringAddr". The number of written characters is shown in the parameter table.

**Note**

**Numerical conversion**

If the numerical converter is used to display information, then it is better to avoid performing a conversion in every PLC cycle for the sake of reducing the PLC cycle time.

The conversion routine can be used independently of the display control. This is to be queried in parameter row "0", although the convert parameter should be set. Consequently, only the string is processed, and the converter routine is executed.

**High display resolution**

If, for example, the axis value is to be displayed with a higher resolution, the following must be observed:

- The variables are read as before with FB2 or FB5. Instead of anypointer BYTE 8 as criterion for output as 64-bit floating point number, REAL 2 is used (e.g.: P#M100.0 REAL 2).
- When specifying the 64-bit floating point number on the HHU/HT 2, you can select the output format with up to 14 places, distributed freely before and after the decimal point, instead of fixed, specified formats.

**HHU output signals**

Byte 1 is used by the HHU output signals and the character specifications are used by the block FC13. These may not be written by the PLC user program.

**Relevant FB1 parameters**

**Handheld unit HHU**

In OB100, the FB1 parameters must be set for the input and output data of the handheld unit:

- Parameter "BHGIn" corresponds to the input data of the PLC from the handheld unit (data received by PLC)
- Parameter "BHGOut" corresponds to the output data of the PLC to the handheld unit (data transmitted by PLC).

The two pointers must be set to the starting point of the relevant data area, which is also parameterized in SDB 210 with an MPI link.

The FB1 parameter "HHU" = 2 must be set for operating an HHU.

**HT 2 handheld terminal**

If HT 2 is used, FB1 parameter "HHU" = 5 must be set. The parameters of the input and output data must be set, as described in the above paragraph "Handheld Unit HHU".

The value that was configured at S2 of the DIP-Fix switch (rotary coding switch) of the connecting module of the HT 2 must be assigned to the parameters "BHGRcGDNo" and "BHGRcGBZNo".
Declaration of the function

DATA_BLOCK "strdat"
STRUCT
  disp: STRING [32]:= 'character_line1 character_line2';
END_STRUCT;
BEGIN
END_DATA_BLOCK

FUNCTION FC13: VOID
VAR INPUT
  Row : BYTE ;
  ChrArray : STRING ;
  Convert : BOOL;
  Addr: POINTER;
  DataType : BYTE ;
  StringAddr : INT;
  Digits : BYTE ;
END VAR
VAR OUTPUT
  Error : BOOL;
END VAR

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row</td>
<td>I</td>
<td>BYTE</td>
<td>0, 1, 2 ... 8, B#16#F</td>
<td>Display line &quot;binary&quot; evaluation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: no display output</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Line 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: Line 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3: Line 1 and line 2 to be changed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4: Line 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5: Line 1 and line 3 to be changed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8: Line 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B#16#F automatic change of all 4 lines</td>
<td></td>
</tr>
<tr>
<td>ChrArray</td>
<td>I</td>
<td>STRING</td>
<td>&quot;DBName&quot;,&lt;VarName&gt;</td>
<td>Display content as pointer to string</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>string[32]: not HT 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>string[64]: HT 2</td>
<td></td>
</tr>
<tr>
<td>Convert</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Activation of numerical conversion</td>
</tr>
<tr>
<td>Addr</td>
<td>I</td>
<td>Pointer</td>
<td></td>
<td>Pointer to the variable to be converted</td>
</tr>
</tbody>
</table>
### Signal Definitions

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataType</td>
<td>I</td>
<td>BYTE</td>
<td>1, 2, 3 ... 8, B#16#13, B#16#30</td>
<td>Data type of the tag 1: BOOL, 1 character 2: BYTE, 3 characters 3: CHAR, 1 character 4: WORD, 5 characters 5: INT, 6 characters 6: DWORD, 7 characters 7: DINT, 8 characters 8: REAL, 9 characters (7 digits plus a sign and a decimal point; for places after the decimal point, refer to the Digits Parameter) B#16#13: String, up to 32/64 characters, &quot;Addr&quot; must be a pointer to a STRING. B#16#30: REAL64, (12 characters: 10 digits plus a sign and a decimal point; for places after the decimal point, refer to the Digits Parameter)</td>
</tr>
<tr>
<td>StringAddr</td>
<td>I</td>
<td>INT</td>
<td>1 ... 32 / 64</td>
<td>Right-justified address within variable from &quot;ChrArray&quot;</td>
</tr>
<tr>
<td>Digits</td>
<td>I</td>
<td>BYTE</td>
<td>1, 2, 3 ... 9</td>
<td>Number of places after the decimal point: 1 ... 4: DataType REAL 1 ... 9: DataType REAL64</td>
</tr>
<tr>
<td>Error</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Error 1: An error occurred</td>
</tr>
</tbody>
</table>

### Ranges of values

#### Ranges of values of data types

<table>
<thead>
<tr>
<th>Data type</th>
<th>Representable numerical range</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>0, 1</td>
</tr>
<tr>
<td>BYTE</td>
<td>0 ... 255</td>
</tr>
<tr>
<td>WORD</td>
<td>0 ... 65535</td>
</tr>
<tr>
<td>INT</td>
<td>-32768 ... 32767</td>
</tr>
<tr>
<td>DWORD</td>
<td>0 ... 9999999</td>
</tr>
<tr>
<td>DINT</td>
<td>-99999999 ... 99999999</td>
</tr>
<tr>
<td>REAL (Digits := 1)</td>
<td>-99999999.9 ... 99999999.9</td>
</tr>
<tr>
<td>REAL (Digits := 2)</td>
<td>-999999.99 ... 999999.99</td>
</tr>
<tr>
<td>REAL (Digits := 3)</td>
<td>-9999.9999 ... 9999.9999</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>REAL (Digits := 9)</td>
<td>-0.99999999 ... 0.99999999</td>
</tr>
</tbody>
</table>
Call example

// DB with the name strdat is declared in symbol table, data element disp is declared as String[32] (in HT 2;
// String[64]) and completely assigned with characters

CALL FC13(  
  Row := MB 26,  
  ChrArray := "strdat".disp,  
  Convert := M 90.1,  
  Addr := P#M 20.0, // Number to be converted  
  DataType := MB 28, // Data type of the variables  
  StringAddr := MW 30,  
  Digits := B#16#3, // 3 decimal places  
  Error := M 90.2);  

13.17.21 FC17: Star-delta - star-delta changeover

Function

Block FC17 is used for star-delta changeover for digital main spindle drives. The changeover can be made in both directions (star > delta or delta > star).

Prerequisite

The prerequisite is two isolated contactors. The contactors are controlled via the peripheral output signals configured at the FC17 outputs: "Y" or "Delta".

Internal sequence

The internal sequence when changing over between star and delta after switching over the FC17 control signal is described in the following: "YDelta" displayed.

1. DB31, ... .DBX21.5 = 0 (reset feedback signal "motor selected")
   DB31, ... .DBX21.x = 1 (set request "2nd motor data set" corresponding to the interface parameterization DB31, ... .DBX130.0 - 4 (Page 65))
2. DB31, ... .DBX93.7 == 0 (feedback signal "pulses enabled" was reset) ⇒
   - Start FC17 timer
   - FC17: "Y" = 0 (reset output for star contactor)
3. After the FC17 timer elapses (FC17: "TimeVal") ⇒
   - FC17: "Y" = 1 (set output for delta contactor)
4. After the FC17 timer elapses again (FC17: "TimeVal") ⇒
   - The pulses are again internally set
   - DB31, ... .DBX93.7 == 1 (feedback signal "pulses enabled" was set) ⇒
   - DB31, ... .DBX21.5 = 0 (feedback signal "motor selected" was set)
Signal flow

1. FC17: Input: Signal for star-delta changeover
2. Changeover: Star > Delta
3. Changeover: Delta > Star
4. FC17: Input: Parameterizable changeover time "TimeVal"
5. Wait time until the control of the output signal: "Y" or "delta"
6. Wait time until pulse enable
7. FC17: Output: Signals for contactor control

References
For additional explanations of motor speed adjustments, see:
- Function Manual, Basic Functions; Spindles (S1); Chapter "Configurable gear adaptation"
- Functions Manual, Basic Functions; Velocities, Setpoint/Actual-Value Systems, Closed-Loop Control (G2)

Note
Drive parameters
The following drive parameters must be considered for a star-delta changeover:

- p833 (dataset changeover configuration)
  - Bit 0 = 1 (contactor switch via application)
  - Bit 1 = 0 (pulse cancellation by drive)
- p826 (motor changeover motor number)
- p827 (motor changeover status word bit number)
General conditions

- Block FC17 must be called absolutely and separately for each spindle.
- The drive pulses are deactivated during a star-delta changeover. The feedback signal to the PLC is realized using:
  - DB31, ... .DBX93.7 == 0 (pulses enabled)
  - DB31, ... .DBX61.7 == 0 (current controller active)
  - DB31, ... .DBX61.6 == 0 (speed controller active)
- For a spindle which is located in an axis mode such as M70 or SPOS, a start-delta changeover is not carried out.
- For a closed-loop position controlled spindle (DB31, ... .DBX61.5 == 1 (position control active)), while the spindle is moving it is not permissible to carry out a star-delta changeover. In the case of a fault, Alarm 25050 "Contour monitoring" is displayed, and the star-delta changeover is not executed.
- Once the star-delta changeover has been initiated using FC17, it cannot be delayed by the user, e.g. by waiting until the star-delta contactors change over during the course of operation. A delay such as this must be implemented by the user in the PLC user program.

Declaration of the function

```plaintext
VAR_INPUT
  YDelta:       BOOL;
  SpindleIFNo:  INT;
  TimeVal:      S5TIME;
  TimerNo:      INT;
END_VAR
VAR_OUTPUT
  Y:            BOOL;
  Delta:        BOOL;
END_VAR
VAR_IN_OUT
  Ref:          WORD;
END_VAR
```

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
</table>
| YDelta:       | I    | BOOL | 0 (FALSE), 1 (TRUE) | Input signal for star-delta changeover
| SpindleIFNo:  | I    | INT  | 1 ...       | Number of the spindle interface (number of the associated machine axis)    |

The changeover is initiated by a signal change:
- 0 (FALSE): Star
- 1 (TRUE): Delta
### Call example

CALL FC17 (  
  YDelta := I 45.7, // Accept star/delta changeover from input 45.7  
  SpindleIFNo := 4, // Spindle interface number: 4  
  // (Number of the associated machine axis)  
  TimeVal := S5T#150ms, // Changeover time: 150 ms  
  TimerNo := 10, // Timer: 10  
  Y := O 52.3, // Control of the star contactor: Output 52.3  
  Delta := O 52.4, // Control of the delta contactor: Output 52.4  
  Ref := MW 50 // Bit memory word 50
);  

### 13.17.22 FC18: SpinCtrl - spindle control

#### Function

Block FC18 (SpinCtrl) can be used to control spindles and axes from the PLC. The block supports the following functions:

- Position spindle
- Rotate spindle
- Oscillate spindle
- Traverse indexing axis
- Traverse positioning axis

Each function is activated by the positive edge of the appropriate initiation signal (start, stop). This signal must remain at a logical "1" until the function has been acknowledged positively or
negatively by InPos="1" or Error = "1". The output parameters are deleted when the relevant trigger signal is reset and the function has been completed.

To be able to control an axis or spindle via the PLC, it must be activated for the PLC. This can, for example, be achieved by calling the block with activation of the "Start" or "Stop" parameter. When you do this, the block requests control of the spindle/axis from the NC.

The NC signals the status of the spindle/axis in the associated axis-specific interface DB31, ... DBX68.4 - 7. Once the axis/spindle is operating under PLC control, the travel command for the active status can be evaluated via the relevant axis-specific interface.

Upon completion ("InPos" is True, "Start" changes to zero), the axis/spindle check function is switched to a neutral status using block FC18.

Alternatively, the PLC user program can also request control for the PLC before calling FC18. By calling this function several times in succession, a better response by the spindle/axis can be obtained as the changeover process in the FC can be omitted.

Activation through the PLC user program is executed in the corresponding spindle interface in byte 8.

After return of the check, the spindle can again be programmed by the NC program.

References

- Function Manual, Basic Functions; Spindles (S1)
- Function Manual, Extended Functions; Positioning Axes (P2)
- Function Manual Expanded Functions; Indexing Axes (T1)

### WARNING

**Changed response behavior of the axis/spindle**

If several block calls (FC18) have been programmed for the same axis/spindle in the PLC user program, then the functions concerned must be interlocked by conditional calls in the user program. The conditional call of a started block (parameter Start or Stop = TRUE) must be called cyclically until the signal state of output parameter "Active" or "InPos" changes from 1 to 0.
**Note**

**Call note**

FC18 must be called cyclically until signal "InPos" or, in the case of an error "Error", produces an edge transition of 1 to 0. Only when the "InPos"/"Error" signal has a 0 state can the axis/spindle concerned be "started" or "stopped" again (the next "start" must be delayed by at least one PLC cycle). This also applies when the assignment in data byte 8 on the axial interface has been changed.

**Cancel**

The function cannot be aborted by means of parameter "Start" or "Stop", but only by means of the axial interface signals (e.g. delete distance-to-go). The axial interface also returns status signals of the axis that may need to be evaluated (e.g. exact stop, traverse command).

**Simultaneity**

Several axes can be traversed simultaneously or subject to a delay by FC18 blocks. The upper limit is only limited by the maximum number of axes of the NC.

**Axis disable**

For a set axis inhibit (DB31, ... .DBX1.4 == 1), the axis controlled via FC18 does not move. Only a simulated actual value is generated. Same behavior as when traversing the axis for an axis inhibit by the NC.

---

**Functions**

**Function 1: Position spindle**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>0 → 1: Start the function</td>
</tr>
<tr>
<td>Funct</td>
<td>1: Function number for &quot;Position spindle&quot;</td>
</tr>
<tr>
<td>Mode</td>
<td>Positioning modes 1, 2, 3, 4 (refer to the paragraph below, &quot;Explanation of the formal parameters&quot;)</td>
</tr>
<tr>
<td>AxisNo</td>
<td>Number of machine axis</td>
</tr>
<tr>
<td>Pos</td>
<td>Position</td>
</tr>
<tr>
<td>FRate</td>
<td>FRate != 0: Positioning velocity FRate = 0: Velocity corresponding to MD35300 $MA_SPIND_POSCTRL_VE- LO</td>
</tr>
<tr>
<td>InPos</td>
<td>1: Position reached with &quot;Exact stop fine&quot;</td>
</tr>
<tr>
<td>Error</td>
<td>1: Positioning error</td>
</tr>
<tr>
<td>State</td>
<td>Error code</td>
</tr>
</tbody>
</table>

**Function 2: Rotate spindle**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>0 → 1: Start the function</td>
</tr>
<tr>
<td>Stop</td>
<td>0 → 1: Stop the function</td>
</tr>
<tr>
<td>Funct</td>
<td>2: Function number for &quot;Rotate spindle&quot;</td>
</tr>
</tbody>
</table>
Function 3: Oscillate spindle

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| Mode :    | Mode = 5: direction of rotation M4  
           | Mode ≠ 5: direction of rotation M3   |
| AxisNo :  | Number of machine axis              |
| FRate :   | Spindle speed                        |
| InPos :   | 1: Setpoint speed is output, also see DB31, ... DBX83.5 (spindle in the set-point range) |
| Error :   | 1: Positioning error                 |
| State :   | Error code                           |

Parameterized oscillation speed: MD35400 $MA_SPIND_OSCILL_DES_VELO

The function of the parameter "Pos" depends on the setting in MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = <value>

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Pos</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0, 1, 2, ... 5</td>
<td>Oscillating</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Oscillation with gear stage change M40</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Oscillation with gear stage change M41</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Oscillation with gear stage change M42</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Oscillation with gear stage change M43</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Oscillation with gear stage change M44</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Oscillation with gear stage change M45</td>
</tr>
</tbody>
</table>

Function 4: Traverse indexing axes

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>The modulo conversion can be compared with approaching the indexing position via POS[AX] = CIC (value) in the part program.</td>
</tr>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>Start :</td>
</tr>
<tr>
<td>Func :</td>
</tr>
<tr>
<td>Mode :</td>
</tr>
<tr>
<td>AxisNo :</td>
</tr>
<tr>
<td>Pos :</td>
</tr>
<tr>
<td>FRate :</td>
</tr>
<tr>
<td>InPos :</td>
</tr>
<tr>
<td>Error :</td>
</tr>
<tr>
<td>State :</td>
</tr>
</tbody>
</table>

Function 5, 6, 7, 8: Position axes

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start :</td>
<td>0 → 1: Start the function</td>
</tr>
<tr>
<td>Func :</td>
<td>5, 6, 7, 8: Function number for &quot;Position axes&quot;</td>
</tr>
<tr>
<td>Mode :</td>
<td>Positioning mode 0, 1, 2, 3, 4</td>
</tr>
<tr>
<td>AxisNo :</td>
<td>Number of machine axis</td>
</tr>
<tr>
<td>Pos :</td>
<td>Position</td>
</tr>
<tr>
<td>FRate :</td>
<td>FRate ≠ 0: Positioning velocity&lt;br&gt;FRate = 0: Velocity corresponding to MD32060 $MA_POS_AX_VEL</td>
</tr>
<tr>
<td>InPos :</td>
<td>1: Position reached with &quot;Exact stop fine&quot;</td>
</tr>
<tr>
<td>Error :</td>
<td>1: Positioning error</td>
</tr>
<tr>
<td>State :</td>
<td>Error code</td>
</tr>
</tbody>
</table>

Function 9: Rotate spindle with automatic gear stage selection:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start :</td>
<td>0 → 1: Start the function</td>
</tr>
<tr>
<td>Stop :</td>
<td>0 → 1: Stop the function</td>
</tr>
<tr>
<td>Func :</td>
<td>9: Function number for &quot;Rotate spindle with gearbox stage selection&quot;</td>
</tr>
<tr>
<td>Mode :</td>
<td>Mode = 5: direction of rotation M4&lt;br&gt;Mode ≠ 5: direction of rotation M3</td>
</tr>
<tr>
<td>AxisNo :</td>
<td>Number of machine axis</td>
</tr>
<tr>
<td>FRate :</td>
<td>Spindle speed</td>
</tr>
<tr>
<td>InPos :</td>
<td>1: Setpoint speed is output</td>
</tr>
<tr>
<td>Error :</td>
<td>1: Positioning error</td>
</tr>
<tr>
<td>State :</td>
<td>Error code</td>
</tr>
</tbody>
</table>
Function 10, 11: Rotate spindle with constant cutting rate

The "Constant cutting rate" function (G96) must be active in the NC.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>0 → 1: Start the function</td>
</tr>
<tr>
<td>Stop</td>
<td>0 → 1: Stop the function</td>
</tr>
</tbody>
</table>
| Funct     | 10: Function number for "Constant cutting rate (m/min)"
|           | 11: Function number for "Constant cutting rate (feet/min)"
| Mode      | Mode = 5: direction of rotation M4
|           | Mode ≠ 5: direction of rotation M3 |
| AxisNo    | Number of machine axis |
| FRate     | Cutting rate |
| InPos     | 1: Setpoint speed is output |
| Error     | 1: Positioning error |
| State     | Error code |

Declaration of the function

FUNCTION FC18: VOID
   //SpinCtrl
   VAR_INPUT
      Start :      BOOL ;
      Stop:       BOOL ;
      Funct:      BYTE ;
      Mode:       BYTE ;
      AxisNo:     INT ;
      Pos:        REAL ;
      FRate:      REAL ;
   END_VAR
   VAR_OUTPUT
      InPos:      BOOL ;
      Error :     BOOL ;
      State :     BYTE ;
   END_VAR

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>0 → 1: Start the function</td>
</tr>
<tr>
<td>Stop</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>0 → 1: Stop the function</td>
</tr>
<tr>
<td>Signal</td>
<td>Type</td>
<td>Type</td>
<td>Value range</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>-------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Funct:  | I    | BYTE  | 1, 2, 3, ... 11 | 1: Position spindle  
|         |      |       |             | 2: Rotate spindle  
|         |      |       |             | 3: Oscillate spindle  
|         |      |       |             | 4: Indexing axis  
|         |      |       |             | 5: Positioning axis metric  
|         |      |       |             | 6: Positioning axis inch  
|         |      |       |             | 7: PosAxis metric with handwheel override  
|         |      |       |             | 8: PosAxis inch with handwheel override  
|         |      |       |             | 9: Rotate spindle with automatic gear stage selection  
|         |      |       |             | 10: Rotate spindle with constant cutting rate (m/min)  
|         |      |       |             | 11: Rotate spindle with constant cutting rate (feet/min)  |
| Mode:   | I    | BYTE  | 0, 1, 2, ... 5 | 0: Positioning to absolute position  
|         |      |       |             | 1: Positioning incremental  
|         |      |       |             | 2: Positioning along the shortest path  
|         |      |       |             | 3: Positioning absolute, positive approach direction  
|         |      |       |             | 4: Positioning absolute, negative approach direction  
|         |      |       |             | 5: Direction of rotation as for M4  |
| AxisNo: | I    | INT   | 1, 2, 3, ... 31 | Number of the axis/spindle to be traversed |
| Pos:    | I    | REAL  | ±0.14693681 -38 to ±0.17014121 +39 | Rotary axis: Degrees  
|         |      |       |             | Indexing axis: Indexing position  
|         |      |       |             | Linear axis: mm or inches |
| FRate:  | I    | REAL  | ±0.14693681 -38 to ±0.17014121 +39 | Rotary axis and spindle: [rev/min]  
|         |      |       |             | Linear axes: [m/min] or [ft/min] |
| InPos:  | O    | BOOL  | 0 (FALSE), 1 (TRUE) | 1: Position reached or function executed |
| Error:  | O    | BOOL  | 0 (FALSE), 1 (TRUE) | 1: Error |
| State:  | O    | BYTE  | 0, 1, 2, ... 255 | Error detection |

**Error identifiers**

An error is active, if: Parameter "Error" == 1 (TRUE)

The cause of the error is displayed in: Parameter "State"

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cause of the error on the PLC side:</td>
</tr>
<tr>
<td>1</td>
<td>Several functions of the axis/spindle were activated simultaneously</td>
</tr>
<tr>
<td>20</td>
<td>A function was started without the position being reached</td>
</tr>
<tr>
<td>30</td>
<td>The axis/spindle was transferred to the NC while still in motion</td>
</tr>
<tr>
<td>State</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>40</td>
<td>The axis is programmed by the NC program, NC internal error</td>
</tr>
<tr>
<td>50</td>
<td>Permanently assigned PLC axis. Traverses (JOG) or is referencing</td>
</tr>
<tr>
<td>60</td>
<td>Permanently assigned PLC axis. Channel status does not permit a start</td>
</tr>
</tbody>
</table>

**Cause of the error on the NC side**

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Incorrect position programmed for axis/spindle (corresponds to alarm 16830)</td>
</tr>
<tr>
<td>101</td>
<td>Programmed speed is too high</td>
</tr>
<tr>
<td>102</td>
<td>Incorrect value range for constant cutting rate (corresponds to alarm 14840)</td>
</tr>
<tr>
<td>104</td>
<td>Following spindle: Illegal programming (corresponds to alarm 22030)</td>
</tr>
<tr>
<td>105</td>
<td>No measuring system available (corresponds to alarm 16770)</td>
</tr>
<tr>
<td>106</td>
<td>Axis positioning still active (corresponds to alarm 22052)</td>
</tr>
<tr>
<td>107</td>
<td>Reference mark not found (corresponds to alarm 22051)</td>
</tr>
<tr>
<td>108</td>
<td>No transition from speed control to position control (corresponds to alarm 22050)</td>
</tr>
<tr>
<td>109</td>
<td>Reference mark not found (corresponds to alarm 22051)</td>
</tr>
<tr>
<td>110</td>
<td>Velocity/speed is negative</td>
</tr>
<tr>
<td>111</td>
<td>Setpoint speed == zero</td>
</tr>
<tr>
<td>112</td>
<td>Invalid gear stage</td>
</tr>
<tr>
<td>115</td>
<td>Programmed position has not been reached</td>
</tr>
<tr>
<td>117</td>
<td>G96/G961 is <strong>not active</strong> in the NC</td>
</tr>
<tr>
<td>118</td>
<td>G96/G961 is <strong>still active</strong> in the NC</td>
</tr>
<tr>
<td>120</td>
<td>Axis is not an indexing axis (corresponds to alarm 20072)</td>
</tr>
<tr>
<td>121</td>
<td>Indexing position error (corresponds to alarm 17510)</td>
</tr>
<tr>
<td>125</td>
<td>DC (shortest distance) not possible (corresponds to alarm 16800)</td>
</tr>
<tr>
<td>126</td>
<td>Absolute value minus not possible (corresponds to alarm 16820)</td>
</tr>
<tr>
<td>127</td>
<td>Absolute value plus not possible (corresponds to alarm 16810)</td>
</tr>
<tr>
<td>128</td>
<td>No transverse axis available for diameter programming (corresponds to alarm 16510)</td>
</tr>
<tr>
<td>130</td>
<td>Software limit switch plus (corresponds to alarm 20070)</td>
</tr>
<tr>
<td>131</td>
<td>Software limit switch minus (corresponds to alarm 20070)</td>
</tr>
<tr>
<td>132</td>
<td>Working area limit plus (corresponds to alarm 20071)</td>
</tr>
<tr>
<td>133</td>
<td>Working area limit minus (corresponds to alarm 20071)</td>
</tr>
<tr>
<td>134</td>
<td>Frame not permitted for indexing axis</td>
</tr>
<tr>
<td>135</td>
<td>Indexing axis with &quot;Hirth toothing&quot; is active (corresponds to alarm 17501)</td>
</tr>
<tr>
<td>136</td>
<td>Indexing axis with &quot;Hirth toothing&quot; is active and axis not referenced (corresponds to alarm 17503)</td>
</tr>
<tr>
<td>137</td>
<td>Spindle operation not possible for transformed spindle/axis (corresponds to alarm 22290)</td>
</tr>
<tr>
<td>138</td>
<td>Axis: Coordinate system-specific working area plus violated (corresponds to alarm 20082)</td>
</tr>
<tr>
<td>139</td>
<td>Axis: Coordinate system-specific working area minus violated (corresponds to alarm 20082)</td>
</tr>
</tbody>
</table>

**System error**

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>corresponds to alarm 450007</td>
</tr>
</tbody>
</table>

Alarm numbers: References Diagnostics Manual
Signal sequence: Normal case

1. PLC user program: Function start using a positive edge: 0 → 1
2. NC: Positive acknowledgment, function executed / position reached
3. PLC user program: Reset after detecting the positive acknowledgment
4. FC18: Reset of the positive acknowledgment

Signal sequence: Error

1. PLC user program: Function start using a positive edge: 0 → 1
2. NC: Negative acknowledgement, error occurred
3. PLC user program: Reset after detecting the negative acknowledgment
4. FC18: Reset of the negative acknowledgment

Call examples

Example 1: Position spindle:

```plaintext
//Positive acknowledgement resets Start:
U M112.0;          //InPos
```
R M 100.0; //Start
//Negative acknowledgement, after error evaluation (state: MB114) reset with T12
start
U M113.0; // Error
U I 6.4; //Key T12
R M 100.0; //Start
//Start with T13
U I 6.3; //Key T13
AN F 112.0; //Restart only when InPos or Error = 0
AN F 113.0;

CALL FC18(
    Start := M100.0,
    Stop := FALSE,
    Funct := B#16#1, //Position spindle
    Mode := B#16#2, //Shortest path
    AxisNo := 5,
    Pos := MD104,
    FRate := MD108,
    InPos := M112.0,
    Error := M113.0,
    State := MB114);

Example 2: Start spindle rotation:

CALL FC18(
    Start := M100.0,
    Stop := FALSE,
    Funct := B#16$q#, //Rotate spindle
    Mode := B#16#5, //Direction of rotation as for M4
    AxisNo := 5,
    Pos := 0.0,
    FRate := MD108,
    InPos := M112.0,
    Error := M113.0,
    State := MB114);

Example 3: Start spindle oscillation

CALL FC18(
    Start := M100.0,
    Stop := FALSE,
    Funct := B#16#3, //Oscillate spindle
    Mode := B#16#0,
    AxisNo := 5,
    Pos := 0.0,
    FRate := MD108,
Example 4: Traverse indexing axis

```plaintext
CALL FC18(
    Start := M100.0,
    Stop := FALSE, //Not used
    Funct := B#16#4, //Traverse indexing axis
    Mode := B#16#0, //Position absolutely
    AxisNo := 4,
    Pos := MD104,   //Default setting in REAL: 1.0; 2.0; ..
    FRate := MD108,
    InPos := M112.0,
    Error := M113.0,
    State := MB114);
```

Example 5: Position axes

```plaintext
CALL FC18(
    Start := M100.0,
    Stop := FALSE, //Not used
    Funct := B#16#5, //Position axes
    Mode := B#16#1, //Position incrementally
    AxisNo := 6,
    Pos := MD104,
    FRate := MD108,
    InPos := M112.0,
    Error := M113.0,
    State := MB114);
```

13.17.23 FC19: MCP_IFM - transfer of MCP signals to interface

Function

The block FC19 (MCP_IFM, M variant, e.g. MCP 483) is used for transferring data from the machine control panel to the NC/PLC interface:

- Modes
- Axis selections
- WCS/MCS switchover commands
- Traversing keys
• Overrides
• Key-operated switch

The following specifications apply to the feed override, axis travel keys and INC keys depending on the active mode or on the coordinate system selected:

• Feed override
  - The feed override is transferred to the interface of the selected channel and to the interface of the axes.
  - The feed override signals are transferred to the NC channel in addition to the "Rapid traverse override" (DBB5) interface byte if the "Feed override for rapid traverse effective" HMI signal is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this HMI signal.

• Machine functions INC and axis travel keys
  - When the MCS is selected, the signals are transferred to the interface of the selected machine axis.
  - When the WCS is selected, the signals are transferred to the geometry axis interface of the parameterized channel.
  - When the system is switched between MCS and WCS, the active axes are generally deselected.

The LEDs on the machine control panel derived from the selections in the acknowledgement. Feedrate and spindle Start/Stop are not transferred to the interface, but output modally as a "FeedHold" or "SpindleHold" signal. The user can link these signals to other signals leading to a feed or spindle stop (this can be implemented, e.g. using the appropriate input signals in FC10: AL_MSG). The associated LEDs are activated at the same time.

If the machine control panel fails, the signals it outputs are preset to zero; this also applies to "FeedHold" and "SpindleHold" output signals.

FC19 or also FC24, FC25, FC26 can be called a multiple number of times in a single PLC cycle. In this case, the first call in the cycle drives the LED displays. Furthermore, all actions of the parameterized block are carried out in the first call. In the following calls, only a reduced level of processing of the channel and mode group interface takes place. The geometry axes are supplied with directional data only in the first block call in the cycle.

Single block processing can be selected/deselected only in the first call in the cycle.

The second machine control panel can be processed if parameter "ModeGroupNo" has been increased by B#16#10. When parameterizing, the mode group number is contained in the lower nibble.

"BAGNo" = 0 or B#16#10 ⇒ mode group signals are not processed.
"ChanNo" = 0 ⇒ no processing of the channel signals.

The INC selections are transferred to the mode group interface. The activation for this specification is done via the DB10.DBX57.0 (INC inputs in BAG area active) through this block once after power up.

Furthermore, two machine control panels can still be handled in parallel by the FC19 block. The block call for the 2nd machine control panel in OB1 cycle must be set after the call of the 1st machine control panel. Support of 2 machine control panels exists to a limited extent in the
machine control panel blocks. Mutual locking of the axis selections for equally assigned axes for 2 machine control panels is not supported.

**Flexible axis configuration**

It is possible to be flexible in the assignment of axis selections or direction keys for machine axis numbers.

Better support is now provided by the MCP blocks for the use of two MCPs, which are to run in parallel, in particular for an application using two channels and two mode groups. Note that the axis-numbers are also specified in the parameterized mode group number of the MCP block in the axis tables of the relevant MCP.

To provide this flexibility, tables for axis numbers are stored in DB10.

For the **first** machine control panel the table starts at byte 8 (symbolic name: MCP1AxisTbl[1..22]) and for the **second** machine control panel starting at byte 32 (symbolic name: MCP2AxisTbl[1..22]) for the second MCP. The machine axis numbers must be entered byte-wise here.

It is permissible to enter a value of 0 in the axis table. Checks are not made to find illegal axis numbers, meaning that false entries can lead to a PLC Stop.

For **FC19**, the **maximum possible number of axis selections** can also be restricted. This upper limit is set for the first Machine control panel in DB10.DBW30 (symbolic name: MCP1MaxAxis) or for the second Machine control panel in DB10.DBW54 (symbolic name: MCP2MaxAxis) set.

The default setting is 0, corresponding to the maximum number of configured axes. The axis numbers and the limit can also be adapted dynamically. Afterwards, a new axis must be selected for FC19. Axis numbers may not be switched over while the axes are traversing the relevant direction keys.

The compatibility mode is preset with axis numbers **1 to 9** for both MCPs and restricted to the configured number of axes.

**Example**

More than nine axes are to be controlled with FC19 using a special application. We recommend that you proceed as follows:

- Reserve free key on MCP.
- Evaluate this key as a flipflop.
- Evaluate the flipflop output as pos. and neg. edge.
- For a positive edge write one set of axis numbers in the axis table (DB10) and switch on LED via this key
- For a negative edge write a different set of axis numbers in the axis table (DB10) and switch off LED via this key.

**Declaration of the function**

FUNCTION FC19: VOID //symbolic name: MCP_IFM
VAR_INPUT
  BAGNo : BYTE ;
  ChanNo: BYTE ;
  SpindleIFNo: BYTE ;
END_VAR

VAR_OUTPUT
  FeedHold : BOOL;
  SpindleHold : BOOL;
END_VAR

BEGIN
END_FUNCTION

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAGNo</td>
<td>I</td>
<td>BYTE</td>
<td>B#16#00 - B#16#0A</td>
<td>1st MCP: Number of mode group to which the mode signals are transferred</td>
</tr>
<tr>
<td>ChanNo</td>
<td>I</td>
<td>BYTE</td>
<td>B#16#10 - B#16#1A</td>
<td>2nd MCP: Number of mode group to which the mode signals are transferred</td>
</tr>
<tr>
<td>SpindleIFNo</td>
<td>I</td>
<td>BYTE</td>
<td>0 - 31 (B#16#1F)</td>
<td>Number of the axis/spindle to which the spindle data is transferred (number of the associated machine axis)</td>
</tr>
<tr>
<td>FeedHold</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Feed stop from MCP, modal</td>
</tr>
<tr>
<td>SpindleHold</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>Spindle stop from MCP, modal</td>
</tr>
</tbody>
</table>

MCP selection signals to the user interface

Table 13-2  Key-operated switch

<table>
<thead>
<tr>
<th>Source: MCP - Switch</th>
<th>Destination: Interface DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position 0</td>
<td>DB10.DBX56.4</td>
</tr>
<tr>
<td>Position 1</td>
<td>DB10.DBX56.5</td>
</tr>
<tr>
<td>Position 2</td>
<td>DB10.DBX56.6</td>
</tr>
<tr>
<td>Position 3</td>
<td>DB10.DBX56.7</td>
</tr>
</tbody>
</table>
### Table 13-3  Operating modes and machine functions

<table>
<thead>
<tr>
<th>Source: MCP - Key</th>
<th>Destination: Interface DB (parameter BAGNo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOMATIC</td>
<td>Display for BAG 1</td>
</tr>
<tr>
<td>MDI</td>
<td>DB11.DBX0.0</td>
</tr>
<tr>
<td>JOG</td>
<td>DB11.DBX0.2</td>
</tr>
<tr>
<td>REPOS</td>
<td>DB11.DBX1.1</td>
</tr>
<tr>
<td>REF</td>
<td>DB11.DBX1.2</td>
</tr>
<tr>
<td>TEACH IN</td>
<td>DB11.DBX1.0</td>
</tr>
<tr>
<td>INC 1 ... 10 000, INC Var.</td>
<td>DB11.DBX2.0 - 2.5</td>
</tr>
</tbody>
</table>

### Table 13-4  Direction keys rapid traverse override

<table>
<thead>
<tr>
<th>Source: MCP - Key</th>
<th>Destination: Interface DB (parameter ChanNo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direction key +</td>
<td>DB21, ..., DBX12.7</td>
</tr>
<tr>
<td>Direction key -</td>
<td>DB21, ..., DBX12.6</td>
</tr>
<tr>
<td>Rapid traverse override</td>
<td>DB21, ..., DBX12.5</td>
</tr>
<tr>
<td>Direction key +</td>
<td>DB21, ..., DBX16.7</td>
</tr>
<tr>
<td>Direction key -</td>
<td>DB21, ..., DBX16.6</td>
</tr>
<tr>
<td>Rapid traverse override</td>
<td>DB21, ..., DBX16.5</td>
</tr>
<tr>
<td>Direction key +</td>
<td>DB21, ..., DBX20.7</td>
</tr>
<tr>
<td>Direction key -</td>
<td>DB21, ..., DBX20.6</td>
</tr>
<tr>
<td>Rapid traverse override</td>
<td>DB21, ..., DBX20.5</td>
</tr>
</tbody>
</table>

### Table 13-5  Override

<table>
<thead>
<tr>
<th>Source: MCP - Switch</th>
<th>Destination: Interface DB (parameter ChanNo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feed override</td>
<td>DB21, ..., DBB4</td>
</tr>
</tbody>
</table>

The transfer is dependent upon the selected axis. The associated interface bits are deleted for axes which are not selected.
Source: MCP - Switch | Destination: Interface DB (all axis DBs)
---|---
Feed override | DB31, ... .DBB0 (selected axis number) The feed override of the 1st MCP is applied to all axes.
Spindle override | DB31, ... .DBB19 (parameter SpindleIFNo)

Table 13-6 Channel signals

| Source: MCP - Keys | Destination: Interface DB (parameter ChanNo) |
---|---|
NC start | DB21, ... .DBX7.1 |
NC stop | DB21, ... .DBX7.3 |
RESET | DB21, ... .DBX7.7 |
Single block | DB21, ... .DBX0.4 |

Table 13-7 Feedrate, spindle signals

| Source: MCP - Keys | Destination: FC output parameters |
---|---|
Feed stop | Parameter: "FeedHold" linked with memory, LEDs are controlled |
Feed enable | |
Spindle stop | Parameter: "SpindleHold" linked with memory, LEDs are controlled |
Spindle enable | |

Checkback signals from user interface for controlling displays

Table 13-8 Operating modes and machine functions

| Destination: MCP - LED | Source: Interface DB (parameter BAGNo) |
---|---|
AUTOMATIC | DB11.DBX6.0 |
MDI | DB11.DBX6.1 |
JOG | DB11.DBX6.2 |
REPOS | DB11.DBX7.1 |
REF | DB11.DBX7.2 |
TEACH IN | DB11.DBX7.0 |

| Destination: MCP - LED | Source: Interface DB (parameter BAGNo) |
---|---|
INC 1 ... 10 000, INC Var. | DB11.DBX8.0 - 8.5 |
Table 13-9 Channel signals

<table>
<thead>
<tr>
<th>Destination: MCP - LED</th>
<th>Source: Interface DB (parameter ChanNo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC start</td>
<td>DB21, ....DBX35.0</td>
</tr>
<tr>
<td>NC stop</td>
<td>DB21, ....DBX35.2 or DB21, ....DBX35.3</td>
</tr>
<tr>
<td>Single block</td>
<td>DB21, ....DBX0.4</td>
</tr>
</tbody>
</table>

Note
Direction key LEDs are controlled by operating the direction keys.
Axis selection and WCS/MCS LEDs are controlled by operating the relevant pushbutton switch.

Call example

```plaintext
CALL FC19( //Machine control panel M variants Signals to interface
  BAGNo := B#16#1,       // Mode group no. 1
  ChanNo := B#16#1,      // Channel no. 1
  SpindleIFNo := B#16#4, // Spindle Interface Number = 4
  FeedHold := m22.0,     // Feed stop signal modal
  SpindleHold := db2.dbx151.0); //Spindle stop modal in
                               //message DB

With these parameter settings, the signals are sent to the first mode group, the first channel and all axes. In addition, the spindle override is transferred in the 4th axis/spindle interface.
The feed hold signal is passed to bit memory 22.0 and the spindle stop signal to data block DB2, data bit 151.0.

Reconnecting the axis selections
To ensure a flexible assignment of the axis selection keys to the appropriate axis or spindle, FC19 needs not be modified or reprogrammed. The axis number simply has to be entered in axis table DB10.DBB8 and followed as required: The axis number simply has to be entered in axis table DB10.DBB8 and followed as required:

Example
The spindle is defined as the 4th machine axis and must be selected via axis key 9.

Solution:
The value 4 must be entered in DB10 byte (8+(9-1)) for the 4th axis.

CALL FC19( // Signals to interface
  BAGNo := B#16#1,       // Mode group no. 1
  ChanNo := B#16#1,      // Channel no. 1
  SpindleIFNo := B#16#4, // Spindle Interface Number = 4
```
FeedHold := m30.0, // Feed stop signal modal
SpindleHold := m30.1); //Spindle stop modal

13.17.24 FC21: transfer PLC NCK data exchange

Function

Block FC21 is used to exchange data between the PLC and NC. The data are immediately transferred when FC21 is called – not waiting until the next basic PLC program cycle starts. The data transfer is activated by calling the block FC21 with parameter "Enable" = 1

Functions
1. Synchronized action signals: PLC → NC channel
2. Synchronized action signals: NC channel → PLC
3. Fast data exchange PLC-NC (read function in NC)
4. Fast data exchange PLC-NC (write function in NC)
5. Update signals to the NC channel
6. Update signals to axes (data byte 2 of the user interface)
7. Update signals to axes (data byte 4 of the user interface)

Declaration of the function

```plaintext
VAR_INPUT
    Enable : BOOL;
    Funct : BYTE ;
    S7Var : ANY ;
    IVar1 : INT ;
    IVar2 : INT ;
END_VAR

VAR_OUTPUT
    Error : BOOL;
    ErrCode : INT ;
END_VAR
```
### Explanation of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable:</td>
<td>I</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>1: Transferring data</td>
</tr>
<tr>
<td>Funct:</td>
<td>I</td>
<td>BYTE 1, 2, 3, ... 7</td>
<td>1: Synchronized actions at channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: Synchronized actions from channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3: Read data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4: Write data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5: Control signals to channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6: Control signals to axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7: Control signals to axis</td>
</tr>
<tr>
<td>S7Var:</td>
<td>I</td>
<td>ANY S7 data storage area</td>
<td>Depends on &quot;Funct&quot;</td>
</tr>
<tr>
<td>IVAR1:</td>
<td>I</td>
<td>INT ---</td>
<td>Depends on &quot;Funct&quot;</td>
</tr>
<tr>
<td>IVAR2:</td>
<td>I</td>
<td>INT ---</td>
<td>Depends on &quot;Funct&quot;</td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>1: An error is active</td>
</tr>
<tr>
<td>ErrCode:</td>
<td>O</td>
<td>INT ---</td>
<td>Depends on &quot;Funct&quot;</td>
</tr>
</tbody>
</table>

### Function 1, 2: Signals synchronized actions to / from Channel

Synchronized actions can be disabled or enabled by the PLC.

The data area is stored on the user interface in DB21, ... DBBB300 ... 307 (to channel) and DB21, ... DBBB308 ... 315 (from channel). The parameter "S7Var" is not evaluated for this function, but must be assigned an actual parameter (see call example). The data are transferred to/from the NC as soon as FC21 is processed.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable:</td>
<td>I</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>1: Transferring data</td>
</tr>
<tr>
<td>Funct:</td>
<td>I</td>
<td>BYTE 1, 2</td>
<td>1: to channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: from channel</td>
</tr>
<tr>
<td>S7Var:</td>
<td>I</td>
<td>ANY S7 data storage area</td>
<td>Not used</td>
</tr>
<tr>
<td>IVAR1:</td>
<td>I</td>
<td>INT 1, 2, ... max. channel number</td>
<td>Channel number</td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>1: An error is active</td>
</tr>
<tr>
<td>ErrCode:</td>
<td>O</td>
<td>INT 1, 10</td>
<td>1: &quot;Funct&quot; invalid</td>
</tr>
</tbody>
</table>

### Call example:

```plaintext
FUNCTION FC100: VOID
VAR_TEMP
    myAny: ANY;
END_VAR
```
BEGIN
NETWORK

//Deactivate synchronized actions with ID3, ID10 and ID31 in NC channel 1:
SYAK: AUF DB21;
     SET;
     S DBX300.//ID3
     2;
     S DBX301.//ID10
     1;
     S DBX303.//ID31
     6;
     L B#16#1;
     T MB11;
     SPA TRAN;

//Synchronized actions from NCK channel 1:
SYVK: L B#16#2;
     T MB11;
TRAN: CALL FC21 ( Enable := M 10.0, //if TRUE, FC 21 active
           Funct := MB 11,
           S7Var := #myAny, //Not used
           IVAR1 := 1, //Channel no.
           IVAR2 := 0,
           Error := M 10.1,
           ErrCode := MW 12);

END_FUNCTION

Function 3, 4: Fast data exchange PLC-NC

General
A separate, internal data area is provided to allow the highspeed exchange of data between the PLC and NC. The size of the internal data field is preset to 4096 bytes. PLC access operations (reading/writing) are realized via FC21. The assignment of this area (structure) must be identically declared in the NC part program and PLC user program.

These data can be accessed from the NC parts program by commands $A_DBB[x], $A_DBW[x], $A_DB[0], $A_DBR[x] (see Parameter Manual System variables).

The concrete address in the data array is specified by a byte offset (0 to 4095) in parameter "IVAR1". In this case, the alignment must be selected according to the data format, i.e. a Dword starts at a 4byte limit and a word at a 2byte limit. Bytes can be located on any chosen offset within the data field, singlebit access operations are not supported and converted to a byte access operation by FC21. Data type information and quantity of data are taken from the ANY parameter, transferred via S7Var.
Without additional programming actions, data consistency is only ensured for 1 and 2 byte access in the NCK and in the PLC. For the 2-byte consistency this is true only for the data type WORD or INT, but not for the data type BYTE.

In the case of longer data types or transfer of arrays which should be transferred consistently, a semaphore byte must be programmed in parameter "IVAR2", which can be used by FC21 to determine the validity or consistency of a block. This handling must be supported by the NC, i.e. in the part program, by writing or deleting the semaphore byte. The semaphore byte is stored in the same data field as the actual user data.

The semaphore byte is identified by a value between 0 and 4095 in "IVAR2".

The PLC reads and describes the semaphore byte via FC21 in the same call, which should transfer the user data. The PLC programmer only needs to set up a semaphore variable. For access from the NC via the parts program, the semaphore feature must be programmed using individual instructions according to the flow chart shown below. The sequence is different for reading and writing variables.

Only individual variables or ARRAYs can be supported directly by the semaphore technique. Structure transfers must be subdivided into individual jobs. The user must ensure data consistency of this structure by programming a semaphore system.

If "IVAR2" = -1 is set, data are transferred without a semaphore.

**Data exchange with semaphore in PLC (schematic of FC21)**

Basic structure in the NC:
Variable value ranges

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Transferring data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4: Write data</td>
</tr>
<tr>
<td>Funct:</td>
<td>I</td>
<td>BYTE</td>
<td>3, 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3: Read data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4: Write data</td>
</tr>
<tr>
<td>S7Var:</td>
<td>I</td>
<td>ANY</td>
<td>S7 data area, except local data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Source/destination data storage area</td>
</tr>
<tr>
<td>IVAR1:</td>
<td>I</td>
<td>INT</td>
<td>0 ... 4095</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Position offset</td>
</tr>
<tr>
<td>IVAR2:</td>
<td>I</td>
<td>INT</td>
<td>-1 ... 4095</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Semaphore byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Transfer without semaphore: -1</td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: An error is active</td>
</tr>
<tr>
<td>ErrCode:</td>
<td>O</td>
<td>INT</td>
<td>20, 21, 22, 23, 24, 25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20: Alignment error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>21: illegal position offset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>22: Illegal semaphore byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>23: No new data to be read</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>24: Cannot write data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>25: Local data parameterized for S7Var</td>
</tr>
</tbody>
</table>

Example 1: Read double word of position offset 4 with semaphore in byte 0 and store in MD100:
- Data type Dword (4 bytes)
- Position offset 4
CALL FC21( 
Enable := M 10.0,          // if TRUE, FC 21 is active 
Funct := B#16#3,            //Read data 
S7Var := P#M 100.0 DWORD 1, 
IVAR1 := 4, 
IVAR2 := 0, 
Error := M 10.1, 
ErrCode := MW12); 

//Enable while 1, until value is read 
R  M10.0; 

NC programming from synchronized actions 
Data transfer from NC to PLC, with data written via synchronized actions; 
Byte 0 serves as the semaphore
ID=1 WHENEVER $A_DBB[0] == 0 DO $A_DBR[4] = $AA_IM[X] $A_DBB[0] = 1 

Data transfer from PLC to NC, with data read via synchronized actions; 
Byte 1 serves as the Semaphore: 

Example 2: Read word of position offset 8 without semaphore and save in MW 104: 
CALL FC21( 
Enable :=M 10.0,          // if TRUE, FC 21 is active 
Funct :=B#16#3,            //Read data 
S7Var :=P#M 104.0 WORD 1, 
IVAR1 :=8, 
IVAR2 :=-1, 
Error :=M 10.1, 
ErrCode :=MW12);
Function 5: Update control signals to channel

The purpose of this function is to transmit important control signals at high speed in between cyclic data transfers. Data bytes 6 and 7 of user interfaces DB21, ... are transferred to the NC. The channel is specified in parameter "IVAR1". This enable, for example, the feed disable, read-in disable to be transferred outside of the PLC cycle.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: Transferring data</td>
</tr>
<tr>
<td>Funct:</td>
<td>I</td>
<td>BYTE</td>
<td>5</td>
<td>5: Control signals to channel</td>
</tr>
<tr>
<td>S7Var:</td>
<td>I</td>
<td>ANY</td>
<td>S7 data storage area</td>
<td>Not used</td>
</tr>
<tr>
<td>IVAR1:</td>
<td>I</td>
<td>INT</td>
<td>1. Max. channel</td>
<td>Channel number</td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: An error is active</td>
</tr>
<tr>
<td>ErrCode:</td>
<td>O</td>
<td>INT</td>
<td>1, 10</td>
<td>1: &quot;Funct&quot; invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10: Channel no. invalid</td>
</tr>
</tbody>
</table>

Function 6: Update control signals to axes

The purpose of function 6 is to transmit important control signals at high speed in between cyclic data transfers. The data byte 2 of application interface DB31, ... is transferred to the NC. The transfer is performed for all activated axes. This allows the controller enable to be transferred outside the PLC cycle, for example.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: Transferring data</td>
</tr>
<tr>
<td>Funct:</td>
<td>I</td>
<td>BYTE</td>
<td>6</td>
<td>6: Control signals to axes</td>
</tr>
<tr>
<td>S7Var:</td>
<td>I</td>
<td>ANY</td>
<td>S7 data storage area</td>
<td>Not used</td>
</tr>
<tr>
<td>IVAR1:</td>
<td>I</td>
<td>INT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: An error is active</td>
</tr>
<tr>
<td>ErrCode:</td>
<td>O</td>
<td>INT</td>
<td>1</td>
<td>1: &quot;Funct&quot; invalid</td>
</tr>
</tbody>
</table>

Function 7: Update control signals to axes

The purpose of function 7 is to transmit important control signals at high speed in between cyclic data transfers. The data byte 4 of application interface DB31, ... is transferred to the NC. The transfer is performed for all activated axes. This enables, for example, the feed stop to be transferred outside the PLC cycle.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: Transferring data</td>
</tr>
<tr>
<td>Funct:</td>
<td>I</td>
<td>BYTE</td>
<td>7</td>
<td>7: Control signals to axes</td>
</tr>
<tr>
<td>S7Var:</td>
<td>I</td>
<td>ANY</td>
<td>S7 data storage area</td>
<td>Not used</td>
</tr>
<tr>
<td>IVAR1:</td>
<td>I</td>
<td>INT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Error:</td>
<td>O</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1: An error is active</td>
</tr>
<tr>
<td>ErrCode:</td>
<td>O</td>
<td>INT</td>
<td>1</td>
<td>1: &quot;Funct&quot; invalid</td>
</tr>
</tbody>
</table>
13.17.25 FC22: TM_DIR - direction selection for tool management

<table>
<thead>
<tr>
<th>NOTICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use</td>
</tr>
<tr>
<td>Block FC22 &quot;TM_DIR&quot; may only be used in conjunction with the tool management.</td>
</tr>
</tbody>
</table>

**Function**

Referred to the location numbers, e.g. a tool magazine or revolver (indexing axis) block FC22 "TM_DIR" supplies the shortest path and direction of motion for positioning, based on the target and current position.

**Outputs**

- Input FC22: "Start" = 1 ⇒ the outputs are cyclically updated.
- Input FC22: "Start" = 0 ⇒ the outputs are undefined.

**Special positioning**

For direction selection with special positioning (input FC22: "Offset" > 0) a new target position is calculated from the target position, the offset for special positioning as well as the number of magazines locations:

New_target position = ( target position - ( special position -1 ) ) neg. MODULO number_of_magazine locations

The new target position corresponds to the location number at which the magazine must be positioned so that the target position requested by the user corresponds to the location number of the special position.

The directional optimization is active both with and without special positioning.

**Call**

The block must be called once with the appropriate parameter settings for each magazine.

**References**

- Further PI services for tool management:
  - FB4: Request PI service (Page 937)
  - FC7: TM_REV - transfer block for tool change with revolver (Page 997)
  - FC8: TM_TRANS - transfer block for tool management (Page 1000)
- Function Manual, Tool Management

**Declaration of the function**

```plaintext
FUNCTION FC22: VOID
// NAME:          TM_DIR
VAR_INPUT
```
Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MagNo:</td>
<td>INT</td>
<td>1, 2, 3, ...</td>
<td>Magazine number</td>
</tr>
<tr>
<td>ReqPos:</td>
<td>INT</td>
<td>1, 2, 3, ...</td>
<td>Target position (magazine location number)</td>
</tr>
<tr>
<td>ActPos:</td>
<td>INT</td>
<td>1, 2, 3, ...</td>
<td>Actual position (magazine location number)</td>
</tr>
<tr>
<td>Offset:</td>
<td>BYTE</td>
<td>0, 1, 2, ...</td>
<td>Offset for special positioning</td>
</tr>
<tr>
<td>Start:</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1 = start of calculation</td>
</tr>
<tr>
<td>Cw:</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1 = Move magazine clockwise</td>
</tr>
<tr>
<td>Ccw:</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1 = Move magazine counterclockwise</td>
</tr>
<tr>
<td>InPos:</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1 = In position</td>
</tr>
<tr>
<td>Diff:</td>
<td>INT</td>
<td>0, 1, 2, ...</td>
<td>Absolute value of the differential path (shortest path)</td>
</tr>
<tr>
<td>Error:</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
<td>1 = error</td>
</tr>
</tbody>
</table>

Call example

CALL FC22(                                    // Tool management direction selection
  // Inputs
  MagNo := 2,                                  // Magazine number
  ReqPos := mw 20,                             // Target position
  ActPos := mw 22,                             // Current position
  Offset := b#16#0,                           // Offset for special positioning
13.17.26 FC24: MCP_IFM2 - transferring MCP signals to the interface

Function

The block FC24 (MCP_IFM2, M variant, e.g. MCP 310) is used for transferring data from the machine control panel to the NC/PLC interface:

- Modes
- Axis selections
- WCS/MCS switchover
- Traversing keys
- Overrides or override simulation signals
- Key-operated switch position

The following specifications apply to the feed override, axis travel keys and INC keys depending on the active mode or on the coordinate system selected:

- **Feed override:**
  - The feed override is transferred to the interface of the selected channel and to the interface of the axes.
  - The feed override signals are transferred to the NC channel in addition to the "Rapid traverse override" (DBB 5) interface byte if the "Feed override for rapid traverse effective" HMI signal is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this HMI signal.

- **Machine functions for INC and axis travel keys:**
  - When the MCS is selected, the signals are transferred to the interface of the selected machine axis.
  - When the WCS is selected, the signals are transferred to the geometry axis interface of the parameterized channel.
  - When the system is switched between MCS and WCS, the active axes are generally deselected.

The associated LEDs on the machine control panel are derived from the acknowledgements from the relevant selections.
Feedrate and spindle Start/Stop are not transferred to the interface, but output modally as a "FeedHold" or "SpindleHold" signal. The user can link these signals to other signals leading to a feed or spindle stop (this can be implemented, e.g. using the appropriate input signals in FC10: AL_MSG). The associated LEDs are activated at the same time.

The **spindle direction** (+, -) is not switched directly either, but made available as output parameter "SpindleDir" permitting, for example, FC18 to be parameterized. A spindle enable signal is also switched via parameter "SpindleHold". One possible method of moving a spindle directly is to preselect it as an axis so that it can be traversed via (axis) direction keys.

If the machine control panel fails, the signals it outputs are preset to zero; this also applies to "FeedHold" and "SpindleHold" output signals.

FC24 or also FC19, FC25, FC26 can be called a multiple number of times in a single PLC cycle. In this case, the first call in the cycle drives the LED displays. Furthermore, all actions of the parameterized block are carried out in the first call. In the following calls, only a reduced level of processing of the channel and mode group interface takes place. The geometry axes are supplied with directional data only in the first block call in the cycle.

Single block processing can be selected/deselected only in the first call in the cycle.

The second machine control panel can be processed if parameter "ModeGroupNo" has been increased by B#16#10. When parameterizing, the HHU number is contained in the lower nibble (lower 4 bits).

"BAGNo" = 0 or B#16#10 ⇒ mode group signals are not processed.

"ChanNo" = 0 ⇒ no processing of the channel signals.

The INC selections are transferred to the mode group interface. The activation for this specification is done via the DB10.DBX57.0 (INC inputs in BAG area active) through this block once after power up.

Furthermore, two machine control panels can be handled in parallel by this block. The block call for the 2nd machine control panel in OB1 cycle must be set after the call of the 1st MCP.

Support for two MCPs is provided in the control panel blocks up to certain limits (support is not provided as standard for mutual interlocking of axis selections with identical assignments on two MCPs).

**Key-operated switch position**

As of software version 4.5 SP2, the key-operated switch signals in the FC24 are also transferred to the user interface (DBX56.5 to 7). This transfer is made independent of whether a keyswitch is mounted on the MCP.

**Note**

For further information see "FC19: MCP_IFM - transfer of MCP signals to interface (Page 1030) ".

---

**Flexible axis configuration**

It is possible to be flexible in the assignment of axis selections or direction keys for machine axis numbers.

Better support is now provided by the MCP blocks for the use of two MCPs, which are to run in parallel, in particular for an application using two channels and two mode groups. Note that
the axis-numbers are also specified in the parameterized mode group number of the MCP block in the axis tables of the relevant MCP.

To provide this flexibility, tables for axis numbers are stored in DB10. For the 1st machine control panel (MCP) the table starts at byte 8 (symbolic name: MCP1AxisTbl[1..22]) and for the second machine control panel (MCP) starting at byte 32 (symbolic name: MCP2AxisTbl[1..22]) for the second MCP. The machine axis numbers must be entered byte by byte here. It is permissible to enter a value of 0 in the axis table. Checks are not made to find illegal axis numbers, meaning that false entries can lead to a PLC Stop.

For FC24, the maximum possible number of axis selections can also be restricted. This upper limit is set for the first machine control panel in DB10.DBW30 (symbolic name: MCP1MaxAxis) or for the second machine control panel in DB10.DBW54 (symbolic name: MCP2MaxAxis) for the respective MCP.

The default setting is 0, corresponding to the maximum number of configured axes. The axis numbers and the limit can also be adapted dynamically. Afterwards, a new axis must be selected for FC24. Axis numbers may not be switched over while the axes are traversing the relevant direction keys. The compatibility mode is preset with axis numbers 1 to 6 for both MCPs and restricted to the configured number of axes.

Declaration of the function

FUNCTION FC24: VOID
  // NAME: MCP_IFM2

VAR_INPUT
  BAGNo : BYTE ;
  ChanNo: BYTE ;
  SpindleIFNo: BYTE ;
END_VAR

VAR_OUTPUT
  FeedHold : BOOL;
  SpindleHold : BOOL;
  SpindleDir: BOOL;
END_VAR

BEGIN
END_FUNCTION

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAGNo</td>
<td>I</td>
<td>BYTE B#16#00 - B#16#0A</td>
<td>1st MCP: Number of mode group to which the mode signals are transferred</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B#16#10 - B#16#1A</td>
<td>2nd MCP: Number of mode group to which the mode signals are transferred</td>
</tr>
</tbody>
</table>
### Call example

```plaintext
CALL FC24(  
  // Slimline machine control panel M variant
  // Signals to interface
  BAGNo := B#16#1,  
  // Mode group no. 1
  ChanNo := B#16#1,  
  // Channel no. 1
  SpindleIFNo := B#16#4,  
  // Spindle Interface Number = 4
  FeedHold := m22.0,  
  // Feed stop signal modal
  SpindleHold := db2.dbx151.0,  
  // Spindle stop modal in message data block
  SpindleDir:= m22.1);  
  // Spindle direction return
```

With these parameter settings, the signals are sent to the first mode group, the first channel and all axes. In addition, the spindle override is transferred in the 4th axis/spindle interface. The feed hold signal is passed to bit memory 22.0 and the spindle stop signal to data block DB2, data bit 151.0. The spindle direction feedback signal supplied via parameter "SpindleDir" can be used as a direction input for an additional FC18 call.

### 13.17.27 FC25: MCP_IFT - transfer of MCP/OP signals to interface

**Function**

The block FC25 (MCP_IFT, T variant, e.g. MCP 483) is used for transferring data from the machine control panel to the NC/PLC interface:

- Modes
- Direction keys of four axes
- WCS/MCS switchover commands
- Overrides
- Key-operated switch
The following specifications apply to the feed override, axis travel keys and INC keys depending on the active mode or on the coordinate system selected:

- **Feed override:**
  - The feed override is transferred to the interface of the selected channel and to the interface of the axes.
  - The feed override signals are transferred to the NC channel in addition to the "Rapid traverse override" (DBB 5) interface byte if the "Feed override for rapid traverse effective" HMI signal is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this HMI signal.

- **Machine functions for INC and axis travel keys:**
  - When the MCS is selected, the signals are transferred to the interface of the selected machine axis.
  - When the WCS is selected, the signals are transferred to the geometry axis interface of the parameterized channel.

The associated LEDs on the machine control panel derived from the acknowledgements of the relevant selections.

Feedrate and spindle Start/Stop are not transferred to the interface, but output modally as a "FeedHold" or "SpindleHold" signal. The user can link these signals to other signals leading to a feed or spindle stop (this can be implemented, e.g. using the appropriate input signals in FC10: Al_MSG). The associated LEDs are activated at the same time.

If the machine control panel fails, the signals it outputs are preset to zero; this also applies to "FeedHold" and "SpindleHold" output signals.

FC25 or also FC19, FC24, FC26 can be called a multiple number of times in a single PLC cycle. In this case, the first call in the cycle drives the LED displays. Furthermore, all actions of the parameterized block are carried out in the first call. In the following calls, only a reduced level of processing of the channel and mode group interface takes place. The geometry axes are supplied with directional data only in the first block call in the cycle.

Single block processing can be selected/deselected only in the first cycle.

The second machine control panel can be processed if parameter "ModeGroupNo" has been increased by B#16#10. When parameterizing, the HHU number is contained in the lower nibble (lower 4 bits).

"BAGNo" = 0 or B#16#10 ⇒ mode group signals are not processed.

"ChanNo" = 0 ⇒ no processing of the channel signals.

### Flexible axis configuration

It is possible to be flexible in the assignment of axis selections or direction keys for machine axis numbers.

Support is now provided by the MCP blocks for the use of two MCPs, which are operated simultaneously, in particular for an application using two channels and two mode groups. The block call for the second machine control panel in OB1 cycle must be set after the call of the 1st MCP. Note that the axis numbers are also specified in the parameterized mode group number of the MCP block in the axis tables of the relevant MCP.
To provide this flexibility, tables for axis numbers are stored in DB10. For the 1st machine control panel (MCP) the table starts at byte 8 (symbolic name: MCP1AxisTbl[1..22]) and for the second machine control panel (MCP) starting at byte 32 (symbolic name: MCP2AxisTbl[1..22]) for the second MCP. The machine axis numbers are entered here bytewise. It is permissible to enter a value of 0 in the axis table. Checks are not made to find illegal axis numbers, meaning that false entries can lead to a PLC Stop.

The restriction of the possible number of axes at FC25 is realized using the 0-values in the axis table. The axis numbers can also be adapted dynamically. During the manual traversing of axes using the direction keys, the axis numbers must not be switched over. The compatibility mode is preset with axis numbers 1 to 4 for both MCPs and restricted to the configured number of axes.

Note
For further information see "FC19: MCP_IFM - transfer of MCP signals to interface (Page 1030) ".

Declaration of the function

```
FUNCTION FC25: VOID
  // NAME: MCP_IFT

  VAR_INPUT
  BAGNo : BYTE ;
  ChanNo: BYTE ;
  SpindleIFNo: BYTE ;
  END_VAR

  VAR_OUTPUT
  FeedHold : BOOL;
  SpindleHold : BOOL;
  END_VAR

  BEGIN
  END_FUNCTION
```

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAGNo</td>
<td>I</td>
<td>BYTE B#16#00 - B#16#0A</td>
<td>1st MCP: Number of mode group to which the mode signals are transferred</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B#16#10 - B#16#1A</td>
<td>2nd MCP: Number of mode group to which the mode signals are transferred</td>
</tr>
<tr>
<td>ChanNo</td>
<td>I</td>
<td>BYTE B#16#00 - B#16#0A</td>
<td>Number of the channel to which the channel signals are sent</td>
</tr>
</tbody>
</table>
### Signal Type, Value range, Description

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpindleIFNo:</td>
<td>I</td>
<td>BYTE B#16#00 - B#16#1F</td>
<td>Number of the axis/spindle to which the spindle data is transferred (number of the associated machine axis)</td>
</tr>
<tr>
<td>FeedHold:</td>
<td>O</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Feed stop from MCP, modal</td>
</tr>
<tr>
<td>SpindleHold:</td>
<td>O</td>
<td>BOOL 0 (FALSE), 1 (TRUE)</td>
<td>Spindle stop from MCP, modal</td>
</tr>
</tbody>
</table>

### Call example

With these example parameter settings, the signals are sent to the first mode group, the first channel and all axes. The spindle override is transferred to the 4th axis/spindle interface. Feed stop is sent to bit memory 22.0 and spindle stop is sent to the data block DB2, DBX151.0.

```
CALL FC25( // Machine control panel T variants
        // Signals to interface
        BAGNo := B#16#1,   // Mode group no. 1
        ChanNo := B#16#1, // Channel no. 1
        SpindleIFNo := B#16#4, // Spindle Interface Number = 4
        FeedHold := m22.0,   // Feed stop signal modal
        SpindleHold := db2.dbx151.0); // Spindle stop modal in message data block
```

### 13.17.28 FC26: HPU_MCP - transfer of HT 8 signals to the interface

#### Declaration of the function

```
FUNCTION FC26: VOID
// NAME: HPU_MCP

VAR_INPUT
    BAGNo : BYTE ;
    ChanNo: BYTE ;
END_VAR

BEGIN
END_FUNCTION
```
Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAGNo</td>
<td>I</td>
<td>BYTE B#16#00 - B#16#0A</td>
<td>1st MCP: Number of mode group to which the mode signals are transferred</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B#16#10 - B#16#1A</td>
<td>2nd MCP: Number of mode group to which the mode signals are transferred</td>
</tr>
<tr>
<td>ChanNo</td>
<td>I</td>
<td>BYTE B#16#00 - B#16#0A</td>
<td>Number of the channel to which the channel signals are transferred</td>
</tr>
</tbody>
</table>

Call examples

Calling FC26 for the first MCP, the first mode group and the first channel of the NC.

```plaintext
CALL FC26( //Machine control panel of the HT 8
    BAGNo := B#16#01, //1.MCP, 1.BAG
    ChanNo := B#16#01); //Channel 1
```

Calling FC26 for the second MCP, the second mode group and the third channel of the NC.

```plaintext
CALL FC26( //Machine control panel of the HT 8
    BAGNo := B#16#12, //2.MCP, 2.BAG
    ChanNo := B#16#03 ); //Channel 3
```

General description of functions

The function FC26 "HPU_MCP (machine control panel signals of the handheld unit HT 8)" transfers the HT 8-specific signals of the following functions between the HT 8-input/output data areas parameterized in function block FB1 (Parameter: MCPxIn and MCPxOut) and the NC/PLC-interface:

- Modes
- Machine function INC
- Coordinate system WCS or MCS
- Axial traverse key
- Axis selection
- Feed override
- Rapid traverse override
- Keyswitch information

Note

Mode switchover through HT 8 and/or HMI

The function FC2 "GP_HP Basic program, cyclic part" transfers the signals of the mode switchover in such a way that an alternative selection from MCP of HT 8 and of the HMI is possible. The transfer of the HMI signals to the NC/PLC interface can also be deactivated in the function block FB1 with the parameter "MMCToIF" = FALSE.

Active axes

Using HT 8 a maximum of 6 axes can be addressed at the same time. The selection of the axes is to be realized by the user/machine manufacturer in the PLC user program.

Flexible axis configuration

The function FC26 enables a flexible assignment of the machine axes to the traversing keys or to the axis selection. 2 tables are available in DB10 for this purpose:

- Machine axis table 1. MCP: DB10.DBB8 to DBB13 (Table of the machine axis number)
  Symbolic name: MCP1AxisTbl[1..22]
- Machine axis table, 2nd MCP: DB10.DBB32 to DBB37 (Table of the machine axis number)
  Symbolic name: MCP2AxisTbl[1..22]

In the tables the axis numbers n (with n = 1, 2, ...) of the active machine axis are to be entered byte-wise. The value 0 must be entered in the unused table locations.

The table length can be specified to the FC26:

- 1st MCP: DB10.DBB30 (upper limit of the machine axis table)
- 2nd MCP: DB10.DBB54 (upper limit of the machine axis table)

A value of 4, for example, means that FC26 takes into account only the first 4 table entries or machine axes. The maximum value for the FC26 is 6. For value 0 or values greater than 6 the maximum value is taken implicitly.

Note

Please note the following constraints:

- A check of the permissible machine axis numbers is not done. Invalid machine axis numbers can lead to a PLC stop.
- The machine axis numbers can be changed dynamically. The table may not be written, if currently a machine axis is being moved via a traversing key.

Transfer of the traversing key signals depending upon the active coordinate system

The traversing key signals for 6 axes lie in the HT 8 input data area below:

- EB n + 2, Bit 0 - Bit 5 (positive traversing direction)
- EB n + 3, Bit 0 - Bit 5 (negative traversing direction)
The switchover of the coordinate system is done via the input signal:

- EB n + 0, Bit 0 (MCS/WCS)

The input signal is evaluated in FC26 with the help of the edge trigger flag. The active coordinate system is shown in the following output signal:

- AB n + 0, Bit 0 (MCS/WCKS) with 0 = MCS, 1 = WCS

In case of active MCS the traversing key signals of the axes 1 - 6 are transferred in the axis-specific interfaces (DB31, ... .DBX4.6 and DBX4.7 (traversing key +/-)) of the axes specified in the machine axis tables (DB10.DBB8 to DBB13 or DBB32 to DBB37).

In case of active WCS it is assumed that the axes 1 - 3 of the machine axis table are geometric axes. For this reason the traversing key signals:

- Of the axes 1 - 3 (EB n + 2 / 3, Bit 0 - Bit 2) are transferred in the interface of the geometric axes in DB21, ... .DBB12 + (n * 4), with n = 0, 1, 2), bit 6 and bit 7 (traversing keys +/-) of the channel specified with the parameter "ChanNo".
  The assignment of the traversing key signals of axes 1, 2 and 3 to the geometric axes 1, 2 and 3 of the channel is permanent and may not be changed.

- Of the axes 4 - 6 (EB n + 2 / 3, Bit 3 - Bit 5) are transferred in the axis-specific interface (DB31, ... .DBX4.6 and DBX4.7 (traversing keys +/-)) of the axes 4 - 6 entered in the machine axis table (DB10.DBB11 to DBB13 or DBB35 to DBB37).

No traversing of machine axes in WCS

In case of active WCS (AB n + 0, Bit 0 = 1) the traversing of the machine axes can be locked. For this, the following output signals are to be set in the PLC user program:

- AB n + 3, Bit 7 = 1 (For WCS: no machine axes)
  Requirement to the FC26, not to transfer any traversing key signals for the machine axes.
  The traversing key signals for the axes 1 - 3 of the machine axis table are transferred to the geometric axes 1 - 3 of the specified channel. The traversing key signals for the axes 4 - 6 of the machine axis table are not transferred.

- AB n + 2, Bit 6 (axes 7 - n selected)
  Requirement to the FC26 not to transfer any traversing key signals, since the axes 1 - 6 of the machine axis table are switched over. The axes 1 - 3 are thus not geometric axes, but instead also machine axes.

Feed override

The value of the HT 8 override switch is transferred as feed override in the channel-specific interface DB21, ... .DBB4 (feedrate override) of the programmed channel (parameter: "ChanNo") and in the axis-specific interfaces DB31, ... .DBB0 (feedrate override) of the axes programmed in the table DB10.DBB8 to DBB13 (machine axis number).

Rapid traverse override

If for the programmed channel (parameter: "ChanNo") the signal DB21, ... .DBX25.3 = 1 (feed override for rapid traverse) is set, the value of the HT 8 override switch is sent as a rapid traverse override to its channel-specific. Interface transferred in DB21, ... .DBB5 (rapid traverse override) and the signal DB21, ... .DBX6.6 = 1 (rapid traverse override active) is also set.
Machine function INC

The HT 8 signals of the machine function INC are transferred differently depending upon the active coordinate system MCS or WCS:

- **Active coordinate system: MCS**
  The selected machine function INC is transferred for all 6 axes in the axis-specific interfaces in DB31, ..., DBX5.0 to DBX5.5 (machine function) of the axes programmed in the table in DB10.DBB8 to DBB13 (machine axis numbers).

- **Active coordinate system: WCS**
  For the axes 1 to 3 the signals of the machine function INC are transferred in the channel-specific interface in DB21, ..., DBX13.0 to DBX13.5 (machine function) of the programmed channel (Parameter: "ChanNo").
  For the axes 4 to 6 the signals of the machine function INC are transferred in the channel-specific interfaces in DB31, ..., DBX5.0 to DBX5.5 (machine function) of the axes programmed in the table in DB10.DBB11 to DBB13 (machine axis numbers).

The selection signals of the INC machine functions are transferred in the mode group-specific interface DB11.DBB 2 + (n * 20), bit 0 to bit 5 (with n = 0, 1, 2, ...). The FC26 informs the NCK about the activation of the mode group-interface for the INC machine functions once after power-up with DB10.DBX57.0 (INC inputs active in the mode group area).

Handwheel selection

The hand-wheel selection signals are evaluated by HMI and transferred to the corresponding NC/PLC interface signals of the machine or geometric axes:

- **Geometry axes**: DB21, ..., DBB 12 + (n * 4), bit 0 to bit 2 (with n = 0, 1, 2)
- **Machine axes**: DB31, ..., DBX4.0 to DBX4.2

Requirement: FB1 parameters: "HWheelMMC" = TRUE

Multiple call in one PLC cycle

FC26 can be called a multiple number of times in a single PLC cycle. Upon the first call in the PLC cycle:

- All actions of the parameterized blocks are executed
- The LED signals are written in the output area
- In case of selected WCS, the traversing key signals of the geometric axes are written
- The signals for the selection and deselection of the individual block are processed

When FC26 is called more often, then the channel and mode group interface are processed at a reduced rate.
Processing of two MCP

If the function FC26 is called twice for two MCP in the cyclic sequence of the PLC program (organization block OB1), the call for the second MCP must be made after the call for the first MCP.

**Note**

If an axis can be traversed from two MCP, then the implementation of a mutual interlocking is the responsibility of the user (machine manufacturer).

Failure of the MCP of the HT 8

In case of failure of the MCP of the HT 8 all the input signals are set to the value 0.

13.17.28.1 Overview of the NC/PLC interface signals of HT 8

Operating modes and machine functions

<table>
<thead>
<tr>
<th>Source: MCP</th>
<th>Destination: Programmed mode group (Parameter BAGNo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOMATIC</td>
<td>DB11.DBX0.0</td>
</tr>
<tr>
<td>MDI</td>
<td>DB11.DBX0.1</td>
</tr>
<tr>
<td>JOG</td>
<td>DB11.DBX0.2</td>
</tr>
<tr>
<td>REPOS</td>
<td>DB11.DBX1.1</td>
</tr>
<tr>
<td>REF</td>
<td>DB11.DBX1.2</td>
</tr>
<tr>
<td>TEACH IN</td>
<td>DB11.DBX1.0</td>
</tr>
<tr>
<td>INC 1 ... 10000, INC Var.</td>
<td>DB11.DBX2.0 - DBX 2.5</td>
</tr>
</tbody>
</table>

Traversing keys and rapid traverse override

<table>
<thead>
<tr>
<th>Source: MCP</th>
<th>Destination: Geometry axis of the prog. channel (Parameter: Chan-No)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traversing key +</td>
<td>DB21, ... .DBX12.7</td>
</tr>
<tr>
<td>Traversing key -</td>
<td>DB21, ... .DBX12.6</td>
</tr>
<tr>
<td>Rapid traverse override</td>
<td>DB21, ... .DBX12.5</td>
</tr>
<tr>
<td>Traversing key +</td>
<td>DB21, ... .DBX16.7</td>
</tr>
<tr>
<td>Traversing key -</td>
<td>DB21, ... .DBX16.6</td>
</tr>
<tr>
<td>Rapid traverse override</td>
<td>DB21, ... .DBX16.5</td>
</tr>
<tr>
<td>Traversing key +</td>
<td>DB21, ... .DBX20.7</td>
</tr>
<tr>
<td>Traversing key -</td>
<td>DB21, ... .DBX20.6</td>
</tr>
<tr>
<td>Rapid traverse override</td>
<td>DB21, ... .DBX20.5</td>
</tr>
</tbody>
</table>
### 13.17 Block descriptions

<table>
<thead>
<tr>
<th>Source: MCP</th>
<th>Destination: Prog. axes corresponding to the table in DB10, DBB8 - 13 (1. MCP) or DBB32 - 37 (2. MCP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traversing key +</td>
<td>DB31,...,DBX4.7</td>
</tr>
<tr>
<td>Traversing key -</td>
<td>DB31,...,DBX4.6</td>
</tr>
<tr>
<td>Rapid traverse override</td>
<td>DB31,...,DBX4.5</td>
</tr>
</tbody>
</table>

#### Override

<table>
<thead>
<tr>
<th>Source: MCP</th>
<th>Destination: Programmed channel (Parameter: ChanNo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feed override</td>
<td>DB21,...,DBB4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source: MCP</th>
<th>Destination: Prog. axes corresponding to the table in DB10, DBB8 - 13 (1. MCP) or DBB32 - 37 (2. MCP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feed override</td>
<td>DB31,...,DBB0</td>
</tr>
</tbody>
</table>

### Channel signals

<table>
<thead>
<tr>
<th>Source: MCP</th>
<th>Destination: Programmed channel (Parameter: ChanNo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC start</td>
<td>DB21,...,DBX7.1</td>
</tr>
<tr>
<td>NC stop</td>
<td>DB21,...,DBX7.3</td>
</tr>
<tr>
<td>RESET</td>
<td>DB21,...,DBX7.7</td>
</tr>
<tr>
<td>Single BLock</td>
<td>DB21,...,DBX0.4</td>
</tr>
</tbody>
</table>

#### 13.17.28.2 Overview of the NC/PLC interface signals of HT 8

#### Operating modes and machine functions

<table>
<thead>
<tr>
<th>Destination: MCP</th>
<th>Source: Interface-DB (parameter BAGNo) Display for BAG 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOMATIC</td>
<td>DB11.DBX6.0</td>
</tr>
<tr>
<td>MDI</td>
<td>DB11.DBX6.1</td>
</tr>
<tr>
<td>JOG</td>
<td>DB11.DBX6.2</td>
</tr>
<tr>
<td>REPOS</td>
<td>DB11.DBX7.1</td>
</tr>
<tr>
<td>REF</td>
<td>DB11.DBX7.2</td>
</tr>
<tr>
<td>TEACH IN</td>
<td>DB11.DBX7.0</td>
</tr>
</tbody>
</table>
13.17.29 FC19, FC24, FC25, FC26 - description of the source code

Task

Machine control panel at the user interface (FC19 M version, FC24 slim-line version, FC25 T version, FC26 HT 8 version).

Associated blocks

DB7, no. of mode groups, channels, axes
DB7, pointer of the machine control panels,
DB8, storage for the next cycle

Resources used

None.

General

Blocks FC19 (M version), FC24 (slim-line version), FC25 (T version) and FC26 (HT 8 version) send the signals of the machine control panel to and from the user interface.

Parameter

- Input parameter "BAGNo"
  The parameter "BAGNo" is used to select the mode group to be processed. The "ModeGroupNo" parameter also selects the number of the machine control panel (Bit 4).

- Input parameter "ChanNo"
  The parameter "ChanNo" is used to select the channel to be processed.

- Input parameter "SpindleFNo" (not FC26)
  The number of the axis/spindle (number of the associated machine axis) in the interface of which the spindle data is sent is specified in the parameter "SpindleFNo". The spindle override is transferred to this spindle interface. The parameters are checked for incorrect parameterization.

- Output parameters "FeedHold" and "SpindleHold" (not FC26)
  The output parameters "FeedHold" and "SpindleHold" are formed from the 4 buttons feed/spindle disable, feed/spindle enable. When the feed/spindle are disabled, the parameter outputs the value "logical 1".

Information for the next cycle is stored in DB8, bytes 0 to 3 or bytes 62 to 65, depending on the machine control panel number. This information is the edge trigger flag, feed value and selected axis number. The blocks are provided with user data via the pointer parameters in DB7 "MCP1In" and "MCP1Out" ("MCP2In" and "MCP2Out"). The pointers are addressed indirectly via a further pointer from the VAR section of DB7 in order to avoid absolute addressing. This additional pointer is determined symbolically in FB1.
Block Description

All 4 components have a similar structure and are classified for the individual subtasks:

- **Input network**
  In the Input network, various parameters are copied to local variables. The machine control signals (user data for input/output area) are also copied over using the various pointers in DB7 (gp_par). These local variables are handled in the block for reasons of efficiency. Some values are initialized for the startup.

- **Global_IN network**
  In the Global_IN network, the MKS/WKS switch-over with edge evaluation, axle selection, direction keys and input overlays are determined for the additional processing in the block. User-specific changes must take place in this part of the program, which are mainly oriented at the axis selection.

- **NC network**
  Only the keyswitch information is copied in Network NC.

- **Network mode group**
  The mode group network transfers the modes of the keys as dynamic signals to the NCK. The INC checkback signals from the NC are stored temporarily for the corresponding LEDs. If the mode group number is 0, this network is not processed. A number that is too large generates the message 401901 or 402501 and changes over after stop.

- **Channel network**
  The functions NC Start, Stop, Reset and single block with corresponding feedback messages are activated in the Channel network. The direction keys of the geometry axes are supplied if a corresponding preselection is made, otherwise they are cleared. If the channel number is 0, this network is not processed. A too large number generates the message 401902 or 402502 and changes over after stop.

- **Spindle network**
  The Spindle network sends the spindle override to the interface parameterized via "SpindleFNo".

- **Axes network**
  The Axes network sends the feed override to the selected axis interface. The direction keys are assigned to the selected axis/spindle. If an axis has been selected previously, the direction information is set to 0.

- **Global_OUT network**
  The output parameters are prepared and the LED signals of the INC machine function are generated in the Global_OUT network.

- **Output network**
  The Output network transfers the output signals of the machine control panel from the VAR_TEMP image to the logical address. The data for the next cycle are also saved.
Axis selection extension

Global_IN network

The Global_IN network must be modified if more than nine axes are selected. If other keys and LEDs are to be used on the machine control panel here, proceed as follows:

1. The command UD DW#16#Value (comment: Clear all axis LEDs for display) deletes all defined LEDs for axis selections. The bit mask is currently processing the nine axis selection LEDs.

2. The command UW W#16# (comment: "Masking all the axis selection buttons") checks whether the direction has changed. The bit string must be adjusted here.

3. The branch destination list (SPL) must be expanded with new jump labels. The new jump labels should be inserted in descending order before label m009. The selection information should be extended for the new jump labels, as described for labels m009 and m008.

Note

The blocks are made available as STL sources if required. But they do not match the current status of the block. Some details of the actual implementation in C have been developed further. For this reason we recommend that you specify your additional requirements for the blocks and that you pass these onto project management via the sales department.

13.17.30 FC1005: AG_SEND - transfers data to Ethernet CP

Function

The AG_SEND function block transfers the data to be transferred via a configured connection to the Ethernet CP. Together with the AG_RECV function block, data exchange can be established with another station via the integrated "CP 840D sl." This station must be configured in STEP 7, "NetPro."

In the basic program, this function is available as a function block FC1005. This is roughly equivalent to function block FC5 in the "SIMATIC_NET_CP" library. The TCP, UDP, and ISO-on-TCP protocols are supported.

Note

Smaller volume of transmittable data with SINUMERIK CP and UDP or ISO-on-TCP protocol

For SINUMERIK CP, only 240 bytes can be transmitted when function block FC1005 and the UDP or ISO-on-TCP protocol are used.

Description of formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT:</td>
<td>I</td>
<td>BOOL</td>
<td>0 (FALSE), 1 (TRUE)</td>
</tr>
<tr>
<td>ID:</td>
<td>I</td>
<td>INT</td>
<td>1, 2, 3 ...16</td>
</tr>
<tr>
<td>LADDR:</td>
<td>I</td>
<td>WORD</td>
<td>-</td>
</tr>
</tbody>
</table>
### 13.17 Block descriptions

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
</table>
| SEND:  | I    | ANY         | Specifies the address and length. The address of the data area alternatively refers to:  
  ● Bit memory address area  
  ● Data block area |
| LEN:   | I    | INT         | TCP: 1, 2, ... 8192  
  UDP: 1, 2, ... 240  
  ISO-on-TCP: 1, 2, ... 240 |
| DONE:  | O    | BOOL        | 0 (FALSE), 1 (TRUE) |
| ERROR: | O    | BOOL        | 0 (FALSE), 1 (TRUE) |
| STATUS:| O    | WORD        | - |

1) Parameter ACT must be TRUE long enough until the following applies: (DONE == 1) OR (ERROR == 1)
2) For SINUMERIK 840D sl: Parameter LADDR := W#16#8110

### Documentation

A detailed module description can be found in:

- SINUMERIK user interfaces: Online help
- SIMATIC Programming Manual: Program blocks for SIMATIC NET S7-CPs  
  Chapter: "Program blocks for Industrial Ethernet" > "Program blocks for open communication services (SEND/RECEIVE interface)" > "AG_SEND / AG_LSEND / AG_SSEND"

#### 13.17.31 FC1006: AG_RECV - receives data from the Ethernet CP

**Function**

The function block AG_RECV receives data transferred via a configured connection from the Ethernet CP. Together with the AG_SEND function block, data exchange can be established with another station via the integrated "CP 840D sl." This station must be configured in STEP 7, "NetPro."

In the basic program, this function is available as a function block FC1006. This is roughly equivalent to function block FC6 in the "SIMATIC_NET_CP" library.

The TCP, UDP, and ISO-on-TCP protocols are supported.

**Note**

**Smaller volume of transmittable data with SINUMERIK CP and UDP or ISO-on-TCP protocol**

For SINUMERIK CP, only 240 bytes can be transmitted when function block FC1006 and the UDP or ISO-on-TCP protocol are used.
Formal parameters

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Value range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID:</td>
<td>I</td>
<td>INT</td>
<td>1, 2 ...16</td>
</tr>
<tr>
<td>LADDR:</td>
<td>I</td>
<td>WORD</td>
<td>-</td>
</tr>
</tbody>
</table>
| RECV:  | I    | ANY         | -          | Specifies the address and length. The address of the data area alternatively refers to:  
- Bit memory address area  
- Data block area  

| NDR : | O    | BOOL        | 0 (FALSE), 1 (TRUE) | 0: Job running  
1: New data accepted |
| ERROR:| O    | BOOL        | 0 (FALSE), 1 (TRUE) | 0: No error  
1: An error is active |
| STATUS: | O | WORD        | -          | Status display |
| LEN:  | O    | INT         | TCP: 1, 2, ... 8192  
UDP: 1, 2, ... 240  
ISO-on-TCP: 1, 2, ... 240 | Number of bytes that are transmitted |

\(^1\) For SINUMERIK 840D sl: Parameter LADDR := W\#16#8110.

Documentation

A detailed module description can be found in:

- SINUMERIK user interfaces: Online help
- SIMATIC Programming Manual: Program blocks for SIMATIC NET S7-CPs  
  Chapter: "Program blocks for Industrial Ethernet" > "Program blocks for open communication services (SEND/RECEIVE interface)" > "AG_RECV / AG_LRECV / AG_SRECV"

13.18 Signal/data descriptions

13.18.1 Interface signals NCK/PLC, HMI/PLC, MCP/PLC

References

The NCK/PLC, HMI/PLC and MCP/PLC interface signals are contained in the Lists document.  
Lists sl (Book2)

The reference code contained therein (according to the signal names) refer to the respective function description, in which the signal is described.

The NCK signals that are evaluated by the basic program and transferred in conditioned form to the user interface are presented in the following sections.
13.18.2 Decoded M signals

The M functions programmed in the part program, ASUB or synchronized actions are channel specifically transferred from the NC to the PLC:

- M functions from channel 1: DB21
- M functions from channel 2: DB22
- etc.

The signal length is one PLC cycle.

Note

The spindle-specific M functions below are not decoded: M3, M4, M5, and M70.

<table>
<thead>
<tr>
<th>Addresses in DB21, ...</th>
<th>Variable</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX194.0 ... 7</td>
<td>M_Fkt_M0 ... M7</td>
<td>BOOL</td>
<td>M signals M0 ... M7</td>
</tr>
<tr>
<td>DBX195.0 ... 7</td>
<td>M_Fkt_M8 ... M15</td>
<td>BOOL</td>
<td>M signals M8 ... M15</td>
</tr>
<tr>
<td>DBX196.0 ... 7</td>
<td>M_Fkt_M16 ... M23</td>
<td>BOOL</td>
<td>M signals M16 ... M23</td>
</tr>
<tr>
<td>DBX197.0 ... 7</td>
<td>M_Fkt_M24 ... M31</td>
<td>BOOL</td>
<td>M signals M24 ... M31</td>
</tr>
<tr>
<td>DBX198.0 ... 7</td>
<td>M_Fkt_M32 ... M39</td>
<td>BOOL</td>
<td>M signals M32 ... M39</td>
</tr>
<tr>
<td>DBX199.0 ... 7</td>
<td>M_Fkt_M40 ... M47</td>
<td>BOOL</td>
<td>M signals M40 ... M47</td>
</tr>
<tr>
<td>DBX200.0 ... 7</td>
<td>M_Fkt_M48 ... M55</td>
<td>BOOL</td>
<td>M signals M48 ... M55</td>
</tr>
<tr>
<td>DBX201.0 ... 7</td>
<td>M_Fkt_M56 ... M63</td>
<td>BOOL</td>
<td>M signals M56 ... M63</td>
</tr>
<tr>
<td>DBX202.0 ... 7</td>
<td>M_Fkt_M64 ... M71</td>
<td>BOOL</td>
<td>M signals M64 ... M71</td>
</tr>
<tr>
<td>DBX203.0 ... 7</td>
<td>M_Fkt_M72 ... M79</td>
<td>BOOL</td>
<td>M signals M72 ... M79</td>
</tr>
<tr>
<td>DBX204.0 ... 7</td>
<td>M_Fkt_M80 ... M87</td>
<td>BOOL</td>
<td>M signals M80 ... M87</td>
</tr>
<tr>
<td>DBX205.0 ... 7</td>
<td>M_Fkt_M88 ... M95</td>
<td>BOOL</td>
<td>M signals M88 ... M95</td>
</tr>
<tr>
<td>DBX206.0 ...3</td>
<td>M_Fkt_M96 ... M99</td>
<td>BOOL</td>
<td>M signals M96 ... M99</td>
</tr>
</tbody>
</table>

Note

The M02/M30 auxiliary function output to the PLC does not state that the part program has been terminated. To determine definitely the end of a part program in the channel, the following interface signal must be evaluated:

DB21, ... .DBX33.5 (M02/M30 active)

The channel status must be RESET. The auxiliary function output could arise from an asynchronous subroutine (ASUB) or a synchronized action and has nothing to do with the real end of the parts program in this case.
13.18.3 G Functions

The M functions programmed in the part program, ASUB or synchronized actions are channel specifically transferred from the NC to the PLC:

- G functions from channel 1: DB21
- G functions from channel 2: DB22
- etc.

The signal length is one PLC cycle.

POWER ON

After POWER ON, the value zero, i.e. active G groups undefined, is specified in the NC/PLC interface for all G groups.

Part program end or abort

After part program end or abort, the last state of the G group is retained.

NC START

After NC-START the values of the 8 G-groups specified in the machine data: MD22510 $NC_ GCODE_GROUPS_TO_PLC are overwritten according to the default setting set via the machine data as well as the values programmed in the part program.

<table>
<thead>
<tr>
<th>Addresses in DB21, ...</th>
<th>Variable</th>
<th>Type</th>
<th>Basic position</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB208</td>
<td>G_FKT_GR_1</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 1</td>
</tr>
<tr>
<td>DB209</td>
<td>G_FKT_GR_2</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 2</td>
</tr>
<tr>
<td>DB210</td>
<td>G_FKT_GR_3</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 3</td>
</tr>
<tr>
<td>DB211</td>
<td>G_FKT_GR_4</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 4</td>
</tr>
<tr>
<td>DB212</td>
<td>G_FKT_GR_5</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 5</td>
</tr>
<tr>
<td>DB213</td>
<td>G_FKT_GR_6</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 6</td>
</tr>
<tr>
<td>DB214</td>
<td>G_FKT_GR_7</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 7</td>
</tr>
<tr>
<td>DB215</td>
<td>G_FKT_GR_8</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 8</td>
</tr>
<tr>
<td>DB216</td>
<td>G_FKT_GR_9</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 9</td>
</tr>
<tr>
<td>DB217</td>
<td>G_FKT_GR_10</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 10</td>
</tr>
<tr>
<td>DB218</td>
<td>G_FKT_GR_11</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 11</td>
</tr>
<tr>
<td>DB219</td>
<td>G_FKT_GR_12</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 12</td>
</tr>
<tr>
<td>DB220</td>
<td>G_FKT_GR_13</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 13</td>
</tr>
<tr>
<td>DB221</td>
<td>G_FKT_GR_14</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 14</td>
</tr>
<tr>
<td>DB222</td>
<td>G_FKT_GR_15</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 15</td>
</tr>
<tr>
<td>DB223</td>
<td>G_FKT_GR_16</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 16</td>
</tr>
<tr>
<td>DB224</td>
<td>G_FKT_GR_17</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 17</td>
</tr>
<tr>
<td>DB225</td>
<td>G_FKT_GR_18</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 18</td>
</tr>
<tr>
<td>DB226</td>
<td>G_FKT_GR_19</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 19</td>
</tr>
<tr>
<td>DB227</td>
<td>G_FKT_GR_20</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 20</td>
</tr>
<tr>
<td>DB228</td>
<td>G_FKT_GR_21</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 21</td>
</tr>
<tr>
<td>DB229</td>
<td>G_FKT_GR_22</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 22</td>
</tr>
</tbody>
</table>
13.18 Signal/data descriptions

<table>
<thead>
<tr>
<th>Addresses in DB21, ...</th>
<th>Variable</th>
<th>Type</th>
<th>Basic position</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBB230</td>
<td>G_FKT_GR_23</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 23</td>
</tr>
<tr>
<td>DBB231</td>
<td>G_FKT_GR_24</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 24</td>
</tr>
<tr>
<td>DBB232</td>
<td>G_FKT_GR_25</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 25</td>
</tr>
<tr>
<td>DBB233</td>
<td>G_FKT_GR_26</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 26</td>
</tr>
<tr>
<td>DBB234</td>
<td>G_FKT_GR_27</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 27</td>
</tr>
<tr>
<td>DBB235</td>
<td>G_FKT_GR_28</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 28</td>
</tr>
<tr>
<td>DBB236</td>
<td>G_FKT_GR_29</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 29</td>
</tr>
<tr>
<td>DBB237</td>
<td>G_FKT_GR_30</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 30</td>
</tr>
<tr>
<td>DBB238</td>
<td>G_FKT_GR_31</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 31</td>
</tr>
<tr>
<td>DBB239</td>
<td>G_FKT_GR_32</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 32</td>
</tr>
<tr>
<td>DBB240</td>
<td>G_FKT_GR_33</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 33</td>
</tr>
<tr>
<td>DBB241</td>
<td>G_FKT_GR_34</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 34</td>
</tr>
<tr>
<td>DBB242</td>
<td>G_FKT_GR_35</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 35</td>
</tr>
<tr>
<td>DBB243</td>
<td>G_FKT_GR_36</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 36</td>
</tr>
<tr>
<td>DBB244</td>
<td>G_FKT_GR_37</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 37</td>
</tr>
<tr>
<td>DBB245</td>
<td>G_FKT_GR_38</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 38</td>
</tr>
<tr>
<td>DBB246</td>
<td>G_FKT_GR_39</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 39</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>DBB271</td>
<td>G_FKT_GR_64</td>
<td>BYTE</td>
<td>0</td>
<td>Active G function of group 64</td>
</tr>
</tbody>
</table>

A complete listing of all the G functions is given in:

References:
Programming Manual Fundamentals; Chapter: "List of G-Functions/Preparatory functions"

13.18.4 Message signals in DB 2

DB2 allows the user to display the messages for individual signals on the operator panel front. As the lists of interface signals show, signals are divided into predefined groups. When a message occurs, disappears or is acknowledged, the number entered in the message number column is transferred to the HMI. Text can be stored in the HMI for each message number.

References:
- Lists sl (Book2), see Section "PLC-Messages (DB2)".
- Startup manual; Chapter "Alarm and message texts"

Note
The number of user areas can be parameterized using FB1.

After the configuration has been modified (FB1: MsgUser) DB2/DB3 must be deleted.
13.19 Programming tips with STEP 7

Some useful tips on programming complex machining sequences in STEP7 are given below. This is essentially handling of the data type POINTER or ANY.

Fundamental tips on the structure of the data type POINTER and ANY see: References:
STEP 7-Manual; Chapter: "Designing user programs" > "Register of CPU and saving of data"

13.19.1 Copying data

Copying variants

For the high-speed copying of data from one DB into another it is recommended

- for larger data quantities to use the system function SFC BLKMOV or SFC FILL, because here a high-speed copying takes place.
- the routine given below is for smaller data quantities, because the supply of ANY parameter to the SFCs consumes additional time.

Example

```plaintext
// DB xx.[AR1] is the source
// DI yy.[AR2] is the destination
AUF DB 100;
//Source DB
LAR1 P#20.0;
//Source start address on data byte 20
AUF DI 101;
//Destination DB
LAR2 P#50.0;
//Destination start address on data byte 50
//AR1, AR2, DB, DI loaded beforehand
L 4;
//Transfer 8 bytes
M001:
L DBW [AR1,P#0.0];
//Copy word-oriented
T DIW [AR2,P#0.0];
+AR1 P#2.0;
+AR2 P#2.0;
TAK;
LOOP M001;
```

13.19.2 ANY and POINTER

The following programming examples show the programming mechanism. They demonstrate how input/output and transit variables (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT) are accessed by data types "POINTER" or "ANY" within an FC or FB. The access operations are described in such a way that a part symbolic method of programming can be used.
13.19.2.1 Use of POINTER and ANY in FC

Function

FC99 has input parameters that are defined as POINTER or ANY. The example shows a body program via which the subcomponents of the POINTER or ANY can be accessed. In this case, the DB parameterized with POINTER or ANY is opened and the address offset stored as a crossarea pointer in address register AR1. Thus allowing access to data elements of variables (generally structures and arrays) that are addressed via the POINTER, ANY.

This access operation is described at the end of the relevant program sequence in the example. With data type ANY, it is also possible to execute a check or branch when the variable is accessed based on the data type and the number of elements involved.

Example

```
FUNCTION FC99: VOID
VAR_INPUT
   Row : BYTE ;
   Convert : BOOL; //Activate numerical conversion
   Addr: POINTER; //Points to variable
   Addr1 : ANY ;
END_VAR
VAR_TEMP
   dbchr : WORD ;
   Number: WORD ;
   type : BYTE ;
END_VAR
BEGIN
  NETWORK
  TITLE = //POINTER
  L       P##Addr;
  LAR1 ;   //Retrieve pointer
  L       W [AR1,P#0.0]; //Retrieve DB number
  T       #dbchr;
  L       D [AR1,P#2.0]; //Offset part of pointer
  LAR1 ;   //ANY
  AUF DB [#dbchr]; //Open DB of variables
  L B [AR1,P#40.0]; //Retrieve byte value using pointer with
                    //address offset 40
                    //ANY
  L       P##Addr1;
  LAR1 ;   //Retrieve ANY
  L       B [AR1,P#1.0]; //Retrieve type
```
13.19.2.2 Use of POINTER and ANY in FB

Function

FB99 has input parameters that are defined as POINTER or ANY.

The example shows a body program via which the subcomponents of the POINTER or ANY can be accessed. In this case, the DB parameterized with POINTER or ANY is opened and the address offset stored as a crossarea pointer in address register AR1, thus allowing access to data elements of variables (generally structures and arrays) that are addressed via the POINTER, ANY.

This access operation is described at the end of the relevant program sequence in the example. With data type ANY, it is also possible to execute a check or branch when the variable is accessed based on the data type and the number of elements involved.

Example

FUNCTIONBLOCK FB99
VAR_INPUT
   Row : BYTE ;
   Convert : BOOL;  //Activate numerical conversion
   Addr: POINTER;  //Points to variable
   Addr1 : ANY ;
END_VAR
VAR_TEMP
   dbchr : WORD ;
   Number: WORD ;
   type : BYTE ;
END_VAR
BEGIN
NETWORK
TITLE =
   //POINTER
   L        P##Addr;
   LAR1 ;  //Retrieve pointer from instance DB
13.19.2.3 POINTER or ANY variable for transfer to FC or FB

POINTER or ANY variable

With version 1 or later of STEP 7 it is possible to define a pointer or ANY in VAR_TEMP. The following two examples show how an ANY can be supplied.

Example 1: Transfer ANY parameter via a selection list to another FB (FC)

Several ANY parameters are defined in an FB (FC). A specific ANY parameter must now be chosen from a selection list for transfer to another FB (FC). This can only be done by means of an ANY in VAR_TEMP. 1 to 4 can be set in parameter "WhichAny" in order to select Addr1 to Addr4.

Note

Address register AR2 is used in the block. However, this address register AR2 is also used for multiinstance DBs. For this reason, this FB should not be declared as multi-instance DB.
VAR_INPUT
WhichAny : INT;
   Addr1 : ANY ; //Observe predetermined order
   Addr2 : ANY ;
   Addr3 : ANY ;
   Addr4 : ANY ;
END_VAR

VAR_TEMP
   dbchr : WORD ;
   Number: WORD ;
   type : BYTE ;
   Temp_addr : ANY ;
END_VAR

BEGIN
NETWORK
TITLE = L WhichAny;
DEC 1;
L P#10.0; //10 bytes per ANY
*I;
LAR2;
L P##Addr1;
+AR2; //Add ANY start addresses
L P##Temp_addr;
LAR1 ; //Retrieve pointer from VAR_TEMP
L DID [AR2,P#0.0]; //Transfer pointer value to VAR_TEMP
T LD [AR1,P#0.0];
L DID [AR2,P#4.0];
T LD [AR1,P#4.0];
L DIW [AR2,P#8.0];
T LW [AR1,P#8.0];

CALL FB101, DB100
   (ANYPAR := #Temp_addr); //ANYPAR is data type ANY

Example 2: Transfer an ANY parameter constructed earlier to another FB (FC)

An ANY parameter that has already been compiled must be transferred to another FB (FC). This can be done only by means of an ANY stored in VAR_TEMP.

FUNCTIONBLOCK FB100
VAR_INPUT
   DBNumber: INT;
   DBOffset : INT;
   Data type: INT;
   Number: INT;

END_VAR
VAR_TEMP
   dbchr : WORD;
   Temp_addr : ANY;
END_VAR
BEGIN
NETWORK
TITLE =
   L   P#Temp_addr;
LAR1 ; //Retrieve pointer from VAR_TEMP
L   B#16#10; //ANY identifier
T   LB [AR1,P#0.0];
L   Data type;
T   LB [AR1,P#1.0];
L   Amount;
T   LW [AR1,P#2.0];
L   DBNumber;
T   LW [AR1,P#4.0];
L   DBOffset;
SLD 3; //Offset is a bit offset
T   LD [AR1,P#6.0];
CALL FB101, DB100
   (ANYPAR := #Temp_addr); //ANYPAR is data type ANY

13.19.3 Multiinstance DB

Function

From Version 2 in STEP 7, you can provide multi-instance enabled FBs, i.e. with multi-instance DBs. The primary characteristic of multiinstance DBs is that a data module can be used for various instances of FBs (see STEP 7 documentation). The quantity structure of the DBs can be optimized this way.
Multi-instance DBs should be activated only when they are actually going to be used since they increase the runtime and code size of the FBs.

Note

When complex programs are implemented in multiinstance enabled FBs that use a pointer and address register, it is important for the programmer to observe certain rules.

With multiinstance DBs, the start address of the variable (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) is transferred with the DI data block register and address register AR2. When variables are accessed within the multiinstance enabled FB, the compiler independently controls the access operation via address register AR2. However, when complex program sections also have to work with address registers in the same FB (e.g. to copy data), then the old contents of AR2 must be saved before the register is changed. The contents of AR2 must be restored to their original state before an instance variable (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) is accessed. The AR2 register of the instance is to be saved most usefully in a local variable (VAR_TEMP).

The command "Load pointer to an instance variable" returns a pointer value from the start of the instance data. To be able to access this variable via a pointer, the offset stored in AR2 must be added.

Example

FUNCTION_BLOCK FB99
VAR_INPUT
    varin: INT;
END_VAR
VAR
    variable1: ARRAY[[0..9]
         of INT;
    variable2: INT;
END_VAR
BEGIN
L   P##variable1; //Pointer at start of ARRAY
     //The value 8500 0010 is now in the accumulator
     //and a cross-area pointer is in the AR2
     //Pointer. If one is to work across areas
     //then, during the addition of these
     //two pointers, an area is to be disabled.
AD  DW#16#00FF_FFFF, //Skipping of an area
LAR1 //Load into AR1
TAR2;
+AR1 AR2; //AR2 instance offset to be added
     //You can now indirectly access the ARRAY
     //of variable 1 via AR1.
L   DIW [AR1, P#0.0]; //E.g. access to the first element
END_FUNCTION_BLOCK
13.19.4 Strings

The STRING data type is required by certain services of the basic program. For this reason, some additional facts about the string structure and general handling procedures for parameter assignments are given below.

Structure of strings

A data of type STRING is generally stored (defined) in a data block. There are two methods of defining a string:

1. Only the data type STRING is assigned to a variable. The STEP7 compiler automatically generates a length of 254 characters.

2. Data type STRING is assigned to a variable together with a string length in square parenthesis (e.g. [32]). With this method, the STEP7 compiler generates a string length corresponding to the input.

Two bytes more than prescribed by the definition are always stored for variables of the STRING data type. The STEP 7 compiler stores the maximum possible number of characters in the 1st byte. The 2nd byte contains the number of characters actually used. Normally, the useful length of the assigned strings is stored by the STEP 7 compiler. The characters (1 byte per character) are then stored from the 3rd byte onwards.

String parameters are generally assigned to blocks of the basic program by means of a POINTER or ANY. Such assignments must generally be made using symbolic programming methods. The data block, which contains the parameterizing string, must be stored in the symbol list. The assignment to the basic program block is then made by means of the symbolic data block name followed by a full stop and the symbolic name of the string variable.

13.19.5 Determining offset addresses for data block structures

Function

Another task, which occurs frequently, is symbolic determination of an offset address within a structured DB, e.g. an ARRAY or STRUCTURE is stored somewhere within the DB. After loading the address register symbolically with the start address, you might like to access the individual elements of the ARRAY or STRUCTURE via an address register. One way of loading the address register symbolically is to use an FC whose input parameter is a pointer. The address of the ARRAY or STRUCTURE is then assigned symbolically to the input parameter of this FC in the program. The program code in the FC now determines the offset address from the input parameter, and passes the offset address in the address register (AR1) to the calling function. Symbolic addressing is thus possible even with indirect access.

Example

FUNCTION FC99: VOID
VAR_INPUT
   Addr: POINTER; //Points to variable
END_VAR
BEGIN
NETWORK
TITLE =
L      P##Addr;
LAR1 ; //Retrieve pointer from Addr
L      D [AR1,P#2.0];  //Offset part of pointer of variable
LAR1 ;
END_FUNCTION

13.19.6 FB calls

Function

For optimizing the execution speeds, it is useful to call all function block calls with many static parameters, such as the blocks FB2, 3, 4, 5, and 7 provided by the basic program when starting with the related instance parameters. When starting (OB100), the preassignment of the parameters must be done, which can then no longer be changed in the cyclic part (OB1). These fixed parameter values are no longer parameterized in the cyclic call, because they have already been written in the Instance DB.

Example: Parameterizing the FB2 with instance DB110

The following example shows how a sensible distribution in OB100 and OB1 component can be implemented.

First, the usual call in the cyclic program is displayed.

CALL FB2, DB110(
  Req := M 100.0,  
  NumVar := 2,      //Read 2 variables
  Addr1 := NCVAR.C1_RP_rpa0_0 
  Line1 := W#16#1  
  Addr2 := NCVAR.C1_RP_rpa0_0 
  Line2 := W#16#2  
  Error := M1.0,   
  NDR := M1.1,    
  State := MW 2,  
  RD1 := P#M 4.0 REAL 1,  
  RD2 := P#M 24.0 REAL 1,
)

The modified version of the program call starts from here. Here the call in OB100 is displayed:

CALL FB2, DB110(
  Req := FALSE,
NumVar := 2, //Read 2 variables
Addr1 := NCVAR.C1_RP_rpa0_0
Line1 : W#16#1
Addr2 := NCVAR.C1_RP_rpa0_0
Line2 : W#16#2
RD1 := P#M 4.0 REAL 1,
RD2 := P#M 24.0 REAL 1,

Here, the call still remaining in OB1 is displayed:

CALL FB2, DB110(
    Req := M0.0,
    Error := M1.0,
    NDR := M1.1,
    State := MW 2,
)

Note
Owing to this measure, a shorter cycle time is achieved in OB1, because the static parameter values need not be copied in the instance DB in each OB1 cycle.

The savings of this variant:
The cyclic copying effort of 3 integer values and 4 ANY parameters with respect to the instance DB, which results from 3 time loading of a constant with a 3-time transfer in the instance data block. In case of each ANY transfer, constants are loaded in the data block 4 times with subsequent transfer.

13.20 Data lists

13.20.1 Machine data

13.20.1.1 Display machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MM_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9032</td>
<td>HMI_MONITOR</td>
<td>Determining the PLC data for HMI monitor information</td>
</tr>
</tbody>
</table>
### 13.20.1.2 NC-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN___</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10100</td>
<td>PLC_CYCLIC_TIMEOUT</td>
<td>Cyclic PLC monitoring time</td>
</tr>
<tr>
<td>14504</td>
<td>MAXNUM_USER_DATA_INT</td>
<td>Number of user data (INT)</td>
</tr>
<tr>
<td>14506</td>
<td>MAXNUM_USER_DATA_HEX</td>
<td>Number of user data (HEX)</td>
</tr>
<tr>
<td>14508</td>
<td>MAXNUM_USER_DATA_FLOAT</td>
<td>Number of user data (FLOAT)</td>
</tr>
<tr>
<td>14510</td>
<td>USER_DATA_INT</td>
<td>User data (INT)</td>
</tr>
<tr>
<td>14512</td>
<td>USER_DATA_HEX</td>
<td>User data (HEX)</td>
</tr>
<tr>
<td>14514</td>
<td>USER_DATA_FLOAT[n]</td>
<td>User data (FLOAT)</td>
</tr>
</tbody>
</table>

**Note**

Machine data in integer/hex format is operated in the NC as DWORD. A machine data in floating point format is managed in the NC as FLOAT (8-byte IEEE). They are stored only in the NC/PLC interface and can be read by the PLC user program from DB20 even when the PLC boots.

### 13.20.1.3 Channel-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC___</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>28150</td>
<td>MM_NUM_VDIVAR_ELEMENTS</td>
<td>Number of elements for writing PLC variables</td>
</tr>
</tbody>
</table>

### 13.20.2 Signals

#### 13.20.2.1 Signals from operator panel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active SINUMERIK operating area</td>
<td>DB19.DB21</td>
<td>DB1900.DBB1</td>
</tr>
<tr>
<td>Current screen number</td>
<td>DB19.DBW24</td>
<td>DB1900.DBW4</td>
</tr>
</tbody>
</table>
13.20 Data lists
14.1 Overview

14.1.1 PLC firmware

The PLC of the SINUMERIK 828D is an integrated PLC based on the SIMATIC S7-200 command set.

The PLC user program is essentially programmed using a Windows PC with the "PLC Programming Tool". In addition, the PLC can be diagnosed and edited via the operator interface of the control. A "Ladder-Add-On-Tool" is available in the control for this purpose.

Note the following special features:

- The PLC user program is completely programmed in ladder logic (LAD).
- A subset of the programming language of the S7-200 is supported.
- When loading to the CPU, in addition to the code for execution, the complete project data (including symbols and comments) is loaded into the control. This means that the control always has the project that matches the currently running PLC user program.
- When loading from the CPU, the complete project data (including symbols and comments) is loaded into the PLC Programming Tool and can be processed/edited using this.
- The user must manage the data and process information according to type. The declared data type must be used consistently each time that the data is accessed.

14.1.2 PLC user interface

The user interface is set-up by the PLC firmware, which also organizes the exchange of all signals and data between the PLC on one side and the NCK and HMI on the other side.

The user interface comprises the parts:

- Data interface with cyclic exchange (see "Data interface (Page 1086)"
- Function interface with function or task-related data exchange (see "Function interface (Page 1100)"

The structured data of these interfaces (retentive and non-retentive) are made available to the user by the firmware by assigning to data blocks: The NC (NCK, tool manager, NC channel, axes, spindles, ...) and the HMI are "Communication partners" of the PLC user program.
Figure 14-1 Overview of the user interface of the PLC 828D
14.1.2.1 Data that are cyclically exchanged

Data which is exchanged between the PLC and NC on one side as well as between the PLC and HMI on the other side.

Data to the PLC are provided by the firmware at the cyclic start of the user program. This ensures, for example, that the signals from the NCK remain constant throughout a cycle.

The firmware transfers data from the PLC to the NKC or HMI at the cycle end of the user program.

Interface PLC ↔ NCK

The cyclic data include, e.g. status signals ("program running", "program interrupted") and control signals (start, stop) and auxiliary and G functions.

Data are structured in signals for:

- Modes
- Channels
- Axes/spindles
- General NCK signals

Interface PLC ↔ HMI

These are signals for:

- Program selection via lists
- Messenger control command
- General signals from/to HMI
- Signals from/to the maintenance planner
- Signals from operator panel (retentive area)
- General selection/status signals from/to HMI (retentive area)

14.1.2.2 Alarms and messages

The user interface in DB1600 offers the option of displaying fault and operating messages on the HMI.

The firmware evaluates the signals that have been entered and sends these as coming and going alarms and messages to the HMI where they are displayed. The HMI manages the fault texts.

Note

A maximum of eight PLC alarms is displayed on the HMI.
14.1.2.3 Retentive data

For the retentive data there are the user data blocks DB9000 - DB9063 and the data area DB1400.DBW0 - DBW127. There the user can store all data that should remain valid after POWER OFF/ON. The retentive data is stored in the non-volatile memory, however, not for data backup.

14.1.2.4 Non-retentive data

Non-retentive data (e.g. bit memories, timers and counters) are deleted every time the control is booted.

14.1.2.5 PLC machine data

The PLC machine data are in the NCK machine data area. At POWER ON, this data is transferred by the PLC firmware into DB4500 of the PLC user interface where it can be evaluated by the PLC user program.

References

SINUMERIK 828D Parameter Manual

14.1.3 PLC key data

The integrated PLC has a program memory of 24000 PLC operations which are completely executed in one fixed PLC cycle.

A maximum of 500 operations can be executed in the INT0 interrupt program that can be optionally used. It is executed servo-synchronous and allows the fastest possible reaction to process events. This is the reason that interrupt-capable PLC I/O modules are not required.

<table>
<thead>
<tr>
<th>Data</th>
<th>Number</th>
<th>Special features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main program (MAIN)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Subprograms (SBRx)</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>Interrupts</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Time controlled interrupt</td>
<td>1</td>
<td>Servo-synchronous interrupt program</td>
</tr>
<tr>
<td>Alarms/messages</td>
<td>248</td>
<td>A maximum of 8 alarms is displayed.</td>
</tr>
<tr>
<td>Bit memory</td>
<td>4096</td>
<td>Non-retentive</td>
</tr>
<tr>
<td>Counters</td>
<td>64</td>
<td>Non-retentive</td>
</tr>
<tr>
<td>Timers, of which:</td>
<td>128</td>
<td>Non-retentive.</td>
</tr>
<tr>
<td>10 ms increment interval</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>100 ms increment interval</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>User data block each with a max. of 512 bytes</td>
<td>64</td>
<td>Address range DB9000 to DB9063</td>
</tr>
<tr>
<td>Data transmission NCK ↔ PLC</td>
<td></td>
<td>Via fixed parameterizable interface.</td>
</tr>
</tbody>
</table>
14.1.4 PLC I/O, fast onboard inputs/outputs

For information on the properties of the rapid onboard inputs/outputs and their response times, see Section "Fast on-board inputs and outputs (Page 1083)".

For information on the I/O modules, the machine control panels as well as the assignment of the onboard inputs/outputs to the PLC, see:

References:
Manual PPU SINUMERIK 828D

14.1.5 PLC Toolbox

14.1.5.1 Star/delta changeover

For star/delta changeover, the following block is provided in the PLC Toolbox:

- StarDelta

**Note**

This block can be used to perform a star/delta changeover - also for 1PH8 spindle motors with SMI connected to a SINAMICS S120.

14.2 Fast on-board inputs and outputs

For digital input and output signals that must be especially quickly processed by the control system, the PPU module has several interfaces that can be used to directly connect signals:

- Connector X242:4 input signals, 4 output signals
- Connector X252:4 input signals, 2 output signals

<table>
<thead>
<tr>
<th>Interface</th>
<th>I/O signals</th>
<th>Addressing</th>
</tr>
</thead>
<tbody>
<tr>
<td>X242</td>
<td>DIN1 ... DIN4</td>
<td>I256.0 ... I256.3</td>
</tr>
<tr>
<td></td>
<td>DOUT1 ... DOU T4</td>
<td>Q256.0 ... Q256.3</td>
</tr>
<tr>
<td>X252</td>
<td>DIN9 ... DIN12</td>
<td>I256.4 ... I256.7</td>
</tr>
<tr>
<td></td>
<td>DOUT9 ... DOU T10</td>
<td>Q256.4 ... Q256.5</td>
</tr>
</tbody>
</table>
**Response times**

This results in the following response times depending on the position control cycle, place of execution and digital inputs/outputs that are used:

<table>
<thead>
<tr>
<th>Position control cycle clock</th>
<th>1.5 ms</th>
<th>3 ms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Processing via:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subprogram SBRx in cyclic operation (MAIN, OB1)</td>
<td>12.5 ms</td>
<td>14 ms</td>
</tr>
<tr>
<td>Digital inputs/outputs: <strong>I/O module PP 72/48</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupt program INT0 (servo synchronous)</td>
<td>3 ms</td>
<td>4.5 ms</td>
</tr>
<tr>
<td>Digital inputs/outputs: <strong>Onboard inputs/outputs</strong> of the PPU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) Signal at the input terminal → processing in the PLC program → signal at the output terminal

---

**Read/write access operations**

Input/output signals are directly read and written at the module input/outputs. For the fastest possible access (servo-synchronous), use of the direct operation commands in the interrupt program INT0 is recommended:

<table>
<thead>
<tr>
<th>Command</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct NO contact</td>
<td>-</td>
</tr>
<tr>
<td>Direct NC contact</td>
<td>-</td>
</tr>
<tr>
<td>Directly set bit value</td>
<td>-(SI)-</td>
</tr>
<tr>
<td>Directly reset bit value</td>
<td>-(RI)-</td>
</tr>
<tr>
<td>Directly assign bit value</td>
<td>-( I )-</td>
</tr>
</tbody>
</table>

---

**Parameterization**

Using machine data, the input/output signals can be exclusively allocated to the NC or the PLC:

- MD10366 $MN_HW_ASSIGN_DIG_FASTIN[ <n> ]
- MD10368 $MN_HW_ASSIGN_DIG_FASTOUT[ <n> ]

with <n>: Index to address the input/output byte (0 = 1. byte, 1 = 2. byte, ...)
14.3 Ladder viewer and ladder add-on

14.3.1 Overview

Function
The essential functions of the ladder viewer and ladder add-on are as follows:

- Ladder viewer: Displays the logic operations of contacts and relays in the PLC user program as ladder diagram (LAD)
- Ladder add-on:
  - Is used to troubleshoot the PLC user program
  - Editing or creating interrupt routines
  - Marshaling data

References
A detailed description of the functionality and using the ladder viewer and ladder add-on can be found in the relevant SINUMERIK Operating Manuals:

- Operating Manual, Universal
- Operating Manual, Turning
- Operating Manual, Milling

in the Chapter, "Ladder viewer and ladder add-on"

14.3.2 Parameterization

Display of addresses
Using NC-specific machine data, the display of addresses in the ladder view can be configured according to SIMATIC S7 300 notation:

MD51230 $MN_ENABLE_LADDER_DB_ADDRESSES = <Value>

<table>
<thead>
<tr>
<th>&lt;Value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address display in SIMATIC S7 200 notation (e.g. Vxxxxx)</td>
</tr>
<tr>
<td>1</td>
<td>Address display in SIMATIC S7 300 notation (e.g. DBxx.DBBxxxx)</td>
</tr>
</tbody>
</table>

Enabling / locking editing
Using the NC-specific machine data, the editing of the interrupt programs INT_100 and INT_101 can be enabled and locked:

MD51231 $MN_ENABLE_LADDER_EDITOR = <Value>

<table>
<thead>
<tr>
<th>&lt;Value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The interrupt programs INT_100 and INT_101 cannot be edited.</td>
</tr>
<tr>
<td>1</td>
<td>Interrupt programs INT_100 and INT_101 can be edited.</td>
</tr>
</tbody>
</table>
14.4 PLC Programming Tool

The "PLC Programming Tool" is a tool for writing PLC user programs in a user-friendly fashion. This is a Windows program and must be installed on a Windows PC. For online access to the control, the PC must be connected to the control via one of the Industrial Ethernet ports, X130 (factory network) or X127 (service interface).

When the "PLC Programming Tool" is called, without specifying an existing project, a new project is created with the default name "Project1." This project can be immediately used to generate the PLC user program, saved under any name, and loaded into the control. Existing projects can be opened in the usual Windows manner.

The online help for the "PLC Programming Tool" can also be called with the "F1" function key, just like in Windows.

References
- SINUMERIK 828D Commissioning Manual, turning and milling
  Section: "Scope of delivery and preconditions" > "Communication with the controller" > "This is how you communicate with the controller via the programming tool"
- Online help for the PLC Programming Tool

14.5 Data interface

Data is cyclically exchanged on one hand between the PLC and NC and and on the other hand between the PLC and HMI. This especially means that the data received from HMI and destined for the NC must be marshaled by the user program in order that these become effective.

**Data to the PLC** are provided by the firmware at the start of the user program cycle. This ensures, for example, that the signals from the NCK remain constant throughout a cycle.

**Data from the PLC** are transferred by the firmware to the NKC or HMI at the end of the user program cycle.

All data of this interface are listed in the manual for SINUMERIK 828D, PPU.
14.5.1 PLC-NCK interface

These cyclic data include, e.g. status signals ("Program running", "Program interrupted"), control signals (Start, Stop) and auxiliary and G functions.

Data are structured in signals for:

- Mode signals
- NC channel signals
- Axis and spindle signals
- General NCK signals
- Fast data exchange PLC-NCK

14.5.1.1 Mode signals

DB3000, 3100

The mode signals specified by the machine control panel or the HMI are transferred to the NCK.

There actual states are signaled to the PLC from the NCK.
14.5.1.2 NC channel signals

DB2500, 3200, 3300, 3500

The signals are structured as follows:

- Control/status signals with normal cyclic transfer, see "Mode signals (Page 1087) ".
- Auxiliary and G functions
  These are entered in the interface DBs in two ways.
  First, they are entered with the change signals.
  The M signals M0 to M99 are additionally decoded and the associated interface bits are set for one cycle.
  For G commands, only the groups selected via machine data are entered in the interface data block.
  The S values are also entered together with the related M signals (M03, M04, M05) in the spindle-specific interface. The axis-specific feedrates are also entered in the appropriate axis-specific interface.
14.5.1.3 Axis and spindle signals

DB3200, 3300, 3700, 3800, 3900

The axis-specific and spindle-specific signals are divided into the following groups:

- Shared axis/spindle signals
- Axis signals
- Spindle signals
- Drive signals

The signals are transferred cyclically with the following exceptions. The exceptions include axial F value, M and S value.

An axial F value is entered via the M, S, F distributor if it is transferred to the PLC during the NC machining process.

The M and S value are also entered via the M, S, F distributor if one or both values requires processing.
14.5.1.4 General NCK signals

**DB2600, 2700, 2800, 2900, 4500, 5300**

- Setpoints to digital/analog inputs/outputs of the NCK
- Actual values from the digital/analog inputs/outputs of the NCK
- Keyswitch and Emergency Stop signals
- Ready and status signals of the NCK

14.5.1.5 Fast data exchange PLC-NCK

**DB4900**

Data block DB4900 with a size of 1024 bytes is used for fast information exchange between the PLC and NCK.

The assignment of the area (structure) must be identically negotiated in the NC part program and PLC user user program.
This data can be accessed from the NC part program using the commands $A\_DBB[x]$, $A\_DBW[x]$, $A\_DBD[x]$ and $A\_DBR[x]$; $0 \leq x \leq 1023$ (see Parameter Manual, System variables).

In this case, the alignment of the data must be selected corresponding to its format, i.e. a Dword starts at a 4-byte limit and a word at a 2-byte limit. Bytes can be located at any offset within the data field.

Data consistency is guaranteed for byte, word and Dword accesses. When transferring several data, the consistency must be guaranteed on the user-side using semaphores, which can be used to detect the validity or consistency of a block.

### 14.5.2 PLC-HMI interface

**DB1700, 1800, 1900**

These signals have already been specified in the diagrams of Chapter PLC-NCK interface (Page 1087).

A reference is again made to what has been stated under Data interface (Page 1086): Data received from the HMI and destined for the NC are not automatically entered into the NC interface range. In fact, these signals and data must be marshaled by the user program.

It involves the following signals:
- Program selection via lists
- Messenger control command
- General signals from/to HMI
- Signals from/to the maintenance planner
- Signals from operator panel (retentive area)
- General selection/status signals from/to HMI (retentive area)

#### 14.5.2.1 Operating area numbers

The number of the active operating area is displayed in: DB1900.DBB1

<table>
<thead>
<tr>
<th>Operating area</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine</td>
<td>201</td>
</tr>
<tr>
<td>Parameters</td>
<td>205</td>
</tr>
<tr>
<td>Programming</td>
<td>203</td>
</tr>
<tr>
<td>Program Manager</td>
<td>202</td>
</tr>
<tr>
<td>Diagnostics</td>
<td>204</td>
</tr>
<tr>
<td>Commissioning</td>
<td>206</td>
</tr>
</tbody>
</table>
14.5.2.2 Screen numbers

The current screen number is displayed in: DB1900.DBW4

Screen number ranges

The following screen number ranges are available:
- JOG, manual machine (Page 1092)
- Reference point approach (Page 1097)
- MDA (Page 1097)
- AUTOMATIC (Page 1097)
- Parameters operating area (Page 1098)
- Program operating area (Page 1099)
- Program manager operating area (Page 1100)
- Diagnostics operating area (Page 1100)

Screen numbers: JOG, manual machine

JOG mode

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turning technology</td>
<td></td>
</tr>
<tr>
<td>Cycle start screen for all screens that can be taken over</td>
<td>81</td>
</tr>
<tr>
<td>Milling technology</td>
<td></td>
</tr>
<tr>
<td>Cycle start screen for all screens that can be taken over</td>
<td>3</td>
</tr>
<tr>
<td>Turning/milling</td>
<td></td>
</tr>
<tr>
<td>Start screen</td>
<td>19</td>
</tr>
<tr>
<td>T,S,M</td>
<td>2</td>
</tr>
<tr>
<td>Set WO</td>
<td>21</td>
</tr>
<tr>
<td>Positioning</td>
<td>4</td>
</tr>
<tr>
<td>Face milling</td>
<td>18</td>
</tr>
<tr>
<td>Stock removal</td>
<td>80</td>
</tr>
<tr>
<td>Cycle start screen for all user screens</td>
<td>91</td>
</tr>
<tr>
<td>General settings</td>
<td>1</td>
</tr>
<tr>
<td>Multi-channel function settings</td>
<td>106</td>
</tr>
<tr>
<td>Collision avoidance settings</td>
<td>107</td>
</tr>
<tr>
<td>Measurement log settings</td>
<td>108</td>
</tr>
<tr>
<td>Swiveling</td>
<td>60</td>
</tr>
<tr>
<td>All G functions</td>
<td>100</td>
</tr>
<tr>
<td>Actual zoom value (MCS/WCS)</td>
<td>101</td>
</tr>
<tr>
<td>Thread synchronizing</td>
<td>102</td>
</tr>
<tr>
<td>Retract</td>
<td>103</td>
</tr>
<tr>
<td>Screen</td>
<td>Number</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>Handwheel</td>
<td>104</td>
</tr>
<tr>
<td>Action synchronization</td>
<td>105</td>
</tr>
</tbody>
</table>

**Turning technology: Workpiece zero**

| Workpiece zero (main menu) | 30     |
| User screen                | 31     |
| User screen                | 34     |
| User screen                | 35     |
| User screen                | 36     |
| User screen                | 37     |
| User screen                | 38     |
| User screen                | 38     |
| Measure edge Z             | 40     |

**Turning technology: Workpiece, measurement**

| Measure tool (main menu)   | 50     |
| Manual X or user screen    | 51     |
| Manual Y                   | 71     |
| Manual Z or user screen    | 52     |
| Zoom or user screen        | 53     |
| User screen                | 54     |
| User screen                | 55     |
| Probe calibration X or user screen | 56 |
| Probe calibration Z or user screen | 57 |
| Automatic length in Z      | 58     |
| Automatic length in Y      | 73     |
| Automatic length in X      | 59     |

**Milling technology: Workpiece zero**

| Workpiece zero (main menu) | 30     |
| Measure edge X             | 5      |
| Measure edge X             | 22     |
| Measure edge Z             | 23     |
| User screen                | 7      |
| Align edge or user screen  | 31     |
| Distance 2 edges or user screen | 32 |
| Right-angled corner        | 33     |
| Any corner or user screen  | 8      |
| Rectangular pocket         | 34     |
| 1 hole or user screen      | 9      |
| 2 holes                    | 35     |
| 3 holes                    | 36     |
| 4 holes                    | 37     |
| Rectangular spigot         | 38     |
| 1 circular spigot or user screen | 39 |
### P4: PLC for SINUMERIK 828D
#### 14.5 Data interface

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 circular spigots</td>
<td>40</td>
</tr>
<tr>
<td>4 circular spigots</td>
<td>41</td>
</tr>
<tr>
<td>Set up level</td>
<td>42</td>
</tr>
<tr>
<td>Probe length calibration or user screen</td>
<td>11</td>
</tr>
<tr>
<td>Probe radius calibration</td>
<td>12</td>
</tr>
</tbody>
</table>

**Milling technology: Workpiece, measurement**

| Measure tool (main menu) | 50 |
| Measure length, manual (with milling tool) or measure length in X, manual (with turning tool) or user screen | 16 |
| Measure length in Y, manual (with turning tool) | 74 |
| Measure length in Z, manual (with turning tool) | 24 |
| Measure diameter, manual or user screen | 17 |
| Measure length, automatic (with milling tool) or measure length in X, automatic (with turning tool) or user screen | 13 |
| Measure length in Y, automatic (with turning tool) | 75 |
| Measure length in Z, automatic (with turning tool) | 25 |
| Measure diameter, automatic or user screen | 14 |
| User screen | 51 |
| Probe calibration or user screen | 15 |
| Fixed point calibration or user screen | 52 |

**RunMyScreens (only for set JobShopIntegration)**

| User screen behind horizontal 1st softkey | 96 |
| User screen behind horizontal 2nd softkey | 98 |
| User screen behind horizontal 3rd softkey | 99 |
| User screen behind horizontal 4th softkey | 94 |
| User screen behind horizontal 5th softkey | 95 |
| User screen behind horizontal 6th softkey | 92 |
| User screen behind horizontal 7th softkey | 97 |
| User screen behind horizontal 8th softkey | 90 |
| User screen behind horizontal 9th softkey | 83 |
| User screen behind horizontal 10th softkey | 82 |
| User screen behind horizontal 11th softkey | 93 |
| User screen behind horizontal 12th softkey | 84 |
| User screen behind horizontal 13th softkey | 85 |
| User screen behind horizontal 14th softkey | 86 |
| User screen behind horizontal 15th softkey | 87 |
| User screen behind horizontal 16th softkey | 88 |
## JOG mode, manual machine

<table>
<thead>
<tr>
<th>Screen</th>
<th>Screen number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Turning/milling</strong></td>
<td></td>
</tr>
<tr>
<td>Taper turning</td>
<td>61</td>
</tr>
<tr>
<td>Angle milling</td>
<td>62</td>
</tr>
<tr>
<td>Stop</td>
<td>63</td>
</tr>
<tr>
<td>Straight line</td>
<td>1300</td>
</tr>
<tr>
<td>Straight all axes</td>
<td>1330</td>
</tr>
<tr>
<td>Straight X alpha</td>
<td>1340</td>
</tr>
<tr>
<td>Straight Z alpha</td>
<td>1350</td>
</tr>
<tr>
<td>Circle</td>
<td>1360</td>
</tr>
<tr>
<td>Drilling</td>
<td>1400</td>
</tr>
<tr>
<td>Center drilling</td>
<td>1410</td>
</tr>
<tr>
<td>Drilling, thread centered</td>
<td>1420</td>
</tr>
<tr>
<td>Drilling, centering</td>
<td>1433</td>
</tr>
<tr>
<td>Drilling, drilling</td>
<td>1434</td>
</tr>
<tr>
<td>Drilling, reaming</td>
<td>1435</td>
</tr>
<tr>
<td>Drilling, boring</td>
<td>1436</td>
</tr>
<tr>
<td>Drilling, deep hole drilling</td>
<td>1440</td>
</tr>
<tr>
<td>Drilling, deep hole drilling 2</td>
<td>1441</td>
</tr>
<tr>
<td>Drilling, tapping</td>
<td>1453</td>
</tr>
<tr>
<td>Drill thread milling</td>
<td>1455</td>
</tr>
<tr>
<td>Positions</td>
<td>1473</td>
</tr>
<tr>
<td>Position row</td>
<td>1474</td>
</tr>
<tr>
<td>Position grid</td>
<td>1477</td>
</tr>
<tr>
<td>Position frame</td>
<td>1478</td>
</tr>
<tr>
<td>Position circle</td>
<td>1475</td>
</tr>
<tr>
<td>Position pitch circle</td>
<td>1479</td>
</tr>
<tr>
<td>Obstacle</td>
<td>1476</td>
</tr>
<tr>
<td>Turning</td>
<td>1500</td>
</tr>
<tr>
<td>Turning, stock removal 1</td>
<td>1513</td>
</tr>
<tr>
<td>Turning, stock removal 2</td>
<td>1514</td>
</tr>
<tr>
<td>Turning, stock removal 3</td>
<td>1515</td>
</tr>
<tr>
<td>Turning, groove 1</td>
<td>1523</td>
</tr>
<tr>
<td>Turning, groove 2</td>
<td>1524</td>
</tr>
<tr>
<td>Turning, groove 3</td>
<td>1525</td>
</tr>
<tr>
<td>Turning, undercut form E</td>
<td>1533</td>
</tr>
<tr>
<td>Turning, undercut form F</td>
<td>1534</td>
</tr>
<tr>
<td>Turning, undercut thread DIN</td>
<td>1535</td>
</tr>
<tr>
<td>Turning, undercut thread</td>
<td>1536</td>
</tr>
<tr>
<td>Turning, thread, longitudinal</td>
<td>1543</td>
</tr>
<tr>
<td>Turning, thread, taper</td>
<td>1544</td>
</tr>
<tr>
<td>Screen</td>
<td>Screen number</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>Turning, thread, facing</td>
<td>1545</td>
</tr>
<tr>
<td>Turning, thread, chain</td>
<td>1546</td>
</tr>
<tr>
<td>Turning, cut-off</td>
<td>1550</td>
</tr>
<tr>
<td>Milling</td>
<td>1600</td>
</tr>
<tr>
<td>Milling, face milling</td>
<td>1610</td>
</tr>
<tr>
<td>Milling, rectangular pocket</td>
<td>1613</td>
</tr>
<tr>
<td>Milling, circular pocket</td>
<td>1614</td>
</tr>
<tr>
<td>Milling, rectangular spigot</td>
<td>1623</td>
</tr>
<tr>
<td>Milling, circular spigot</td>
<td>1624</td>
</tr>
<tr>
<td>Milling, longitudinal groove</td>
<td>1633</td>
</tr>
<tr>
<td>Milling, circumferential groove</td>
<td>1634</td>
</tr>
<tr>
<td>Milling, open groove</td>
<td>1635</td>
</tr>
<tr>
<td>Milling, multi-edge</td>
<td>1640</td>
</tr>
<tr>
<td>Milling, thread milling</td>
<td>1454</td>
</tr>
<tr>
<td>Milling, engraving</td>
<td>1670</td>
</tr>
<tr>
<td>Contour turning</td>
<td>1200</td>
</tr>
<tr>
<td>Contour turning, new contour / last contour</td>
<td>1210</td>
</tr>
<tr>
<td>Contour turning, stock removal along contour</td>
<td>1220</td>
</tr>
<tr>
<td>Contour turning, contour grooving</td>
<td>1230</td>
</tr>
<tr>
<td>Contour turning, contour plunge turning</td>
<td>1240</td>
</tr>
<tr>
<td>Contour milling</td>
<td>1100</td>
</tr>
<tr>
<td>Contour milling, new contour / last contour</td>
<td>1110</td>
</tr>
<tr>
<td>Contour milling, path milling</td>
<td>1120</td>
</tr>
<tr>
<td>Contour milling, centering</td>
<td>1130</td>
</tr>
<tr>
<td>Contour milling, rough drilling</td>
<td>1140</td>
</tr>
<tr>
<td>Contour milling, contour pocket</td>
<td>1150</td>
</tr>
</tbody>
</table>

**Turning technology: Simulation**

| Side view | 1740 |
| Front view | 1750 |
| 3D view | 1760 |
| 2-window view | 1770 |
| Half section | 1780 |

**Turning technology: Simultaneous recording**

| Side view | 1741 |
| Front view | 1751 |
| 3D view | 1761 |
| 2-window view | 1771 |
| Machine space | 1791 |
| Half section | 1781 |

**Milling technology: Simulation**

| Top view | 1742 |
| 3D view | 1760 |
### DB19.DBB24

<table>
<thead>
<tr>
<th>Screen</th>
<th>Screen number</th>
</tr>
</thead>
<tbody>
<tr>
<td>From the front</td>
<td>1744</td>
</tr>
<tr>
<td>From the rear</td>
<td>1746</td>
</tr>
<tr>
<td>From the Left</td>
<td>1748</td>
</tr>
<tr>
<td>From the right</td>
<td>1752</td>
</tr>
<tr>
<td>Half section</td>
<td>1780</td>
</tr>
<tr>
<td>Turning view</td>
<td>1782</td>
</tr>
</tbody>
</table>

**Milling technology: Simultaneous recording**

<table>
<thead>
<tr>
<th>Screen</th>
<th>Screen number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top view</td>
<td>1743</td>
</tr>
<tr>
<td>3D view</td>
<td>1761</td>
</tr>
<tr>
<td>From the front</td>
<td>1745</td>
</tr>
<tr>
<td>From the rear</td>
<td>1747</td>
</tr>
<tr>
<td>From the Left</td>
<td>1749</td>
</tr>
<tr>
<td>From the right</td>
<td>1753</td>
</tr>
<tr>
<td>Machine space</td>
<td>1791</td>
</tr>
<tr>
<td>Half section</td>
<td>1781</td>
</tr>
<tr>
<td>Turning view</td>
<td>1783</td>
</tr>
</tbody>
</table>

### Screen numbers: Reference point approach

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual zoom value MCS/WCS</td>
<td>101</td>
</tr>
</tbody>
</table>

### Screen numbers: MDA

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDA</td>
<td>20</td>
</tr>
<tr>
<td>All G functions</td>
<td>100</td>
</tr>
<tr>
<td>Actual zoom value MCS/WCS</td>
<td>101</td>
</tr>
<tr>
<td>Handwheel</td>
<td>104</td>
</tr>
<tr>
<td>Action synchronization</td>
<td>105</td>
</tr>
<tr>
<td>Program control</td>
<td>210</td>
</tr>
<tr>
<td>Settings</td>
<td>250</td>
</tr>
</tbody>
</table>

### Screen numbers: AUTOMATIC

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>200</td>
</tr>
<tr>
<td>Overstore</td>
<td>202</td>
</tr>
<tr>
<td>Program control</td>
<td>210</td>
</tr>
<tr>
<td>Block search</td>
<td>220</td>
</tr>
<tr>
<td>General settings</td>
<td>250</td>
</tr>
<tr>
<td>Screen</td>
<td>Number</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>Multi-channel function settings</td>
<td>106</td>
</tr>
<tr>
<td>Collision avoidance settings</td>
<td>107</td>
</tr>
<tr>
<td>All G functions</td>
<td>100</td>
</tr>
<tr>
<td>Actual zoom value MCS/WCS</td>
<td>101</td>
</tr>
<tr>
<td>Handwheel</td>
<td>104</td>
</tr>
<tr>
<td>Action synchronization</td>
<td>105</td>
</tr>
</tbody>
</table>

**Turning technology: Simultaneous recording**

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Side view</td>
<td>243</td>
</tr>
<tr>
<td>Front view</td>
<td>244</td>
</tr>
<tr>
<td>3D view</td>
<td>245</td>
</tr>
<tr>
<td>2-window view</td>
<td>246</td>
</tr>
<tr>
<td>Machine space</td>
<td>247</td>
</tr>
<tr>
<td>Half section</td>
<td>253</td>
</tr>
</tbody>
</table>

**Milling technology: Simultaneous recording**

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top view</td>
<td>242</td>
</tr>
<tr>
<td>3D view</td>
<td>244</td>
</tr>
<tr>
<td>From the front</td>
<td>248</td>
</tr>
<tr>
<td>From the rear</td>
<td>249</td>
</tr>
<tr>
<td>From the Left</td>
<td>251</td>
</tr>
<tr>
<td>From the right</td>
<td>252</td>
</tr>
<tr>
<td>Machine space</td>
<td>247</td>
</tr>
<tr>
<td>Half section</td>
<td>253</td>
</tr>
<tr>
<td>Turning view</td>
<td>254</td>
</tr>
</tbody>
</table>

**Screen numbers: Parameters operating area**

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool list</td>
<td>600</td>
</tr>
<tr>
<td>Tool wear</td>
<td>610</td>
</tr>
<tr>
<td>User tool list</td>
<td>620</td>
</tr>
<tr>
<td>Magazine</td>
<td>630</td>
</tr>
</tbody>
</table>

**Work offset**

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work offset, active</td>
<td>642</td>
</tr>
<tr>
<td>Work offset, overview</td>
<td>643</td>
</tr>
<tr>
<td>Work offset, basic</td>
<td>644</td>
</tr>
<tr>
<td>Work offset, G54 - G509</td>
<td>645</td>
</tr>
<tr>
<td>Details of work offset, active, overview, basic or G54 - G509</td>
<td>647</td>
</tr>
</tbody>
</table>

**User variable**

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>R parameters</td>
<td>650</td>
</tr>
<tr>
<td>Global GUD 1 (SGUD)</td>
<td>660</td>
</tr>
<tr>
<td>Global GUD 2 (MGUD)</td>
<td>661</td>
</tr>
<tr>
<td>Global GUD 3 (UGUD)</td>
<td>662</td>
</tr>
</tbody>
</table>
### Screen numbers: Program operating area

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turning technology: Simulation</td>
<td></td>
</tr>
<tr>
<td>Side view</td>
<td>413</td>
</tr>
<tr>
<td>Front view</td>
<td>414</td>
</tr>
<tr>
<td>3D view</td>
<td>415</td>
</tr>
<tr>
<td>2-window view</td>
<td>416</td>
</tr>
<tr>
<td>Half section</td>
<td>423</td>
</tr>
<tr>
<td>Milling technology: Simulation</td>
<td></td>
</tr>
<tr>
<td>Top view</td>
<td>412</td>
</tr>
<tr>
<td>3D view</td>
<td>414</td>
</tr>
</tbody>
</table>

### Setting data

<table>
<thead>
<tr>
<th></th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working area limitation</td>
<td>671</td>
</tr>
<tr>
<td>Spindle data</td>
<td>670</td>
</tr>
<tr>
<td>Spindle chuck data</td>
<td>672</td>
</tr>
</tbody>
</table>

### Ctrl-Energy

<table>
<thead>
<tr>
<th>Ctrl-Energy</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl-Energy, main menu</td>
<td>6170</td>
</tr>
<tr>
<td>Ctrl-Energy, analysis</td>
<td>6171</td>
</tr>
<tr>
<td>Ctrl-Energy, profiles</td>
<td>6172</td>
</tr>
<tr>
<td>Ctrl-Energy, analysis graphic</td>
<td>6176</td>
</tr>
<tr>
<td>Ctrl-Energy, analysis long-term measurement</td>
<td>6177</td>
</tr>
<tr>
<td>Ctrl-Energy, analysis details</td>
<td>6179</td>
</tr>
<tr>
<td>Ctrl-Energy, compare measurements</td>
<td>6178</td>
</tr>
</tbody>
</table>
### Screen numbers: Program manager operating area

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC directory</td>
<td>300</td>
</tr>
<tr>
<td>Local drive</td>
<td>325</td>
</tr>
<tr>
<td>USB / configured drive1</td>
<td>330</td>
</tr>
<tr>
<td>Configured drive2</td>
<td>340</td>
</tr>
<tr>
<td>Configured drive3</td>
<td>350</td>
</tr>
<tr>
<td>Configured drive4</td>
<td>360</td>
</tr>
<tr>
<td>Configured drive5</td>
<td>383</td>
</tr>
<tr>
<td>Configured drive6</td>
<td>384</td>
</tr>
<tr>
<td>Configured drive7</td>
<td>385</td>
</tr>
<tr>
<td>Configured drive8</td>
<td>386</td>
</tr>
</tbody>
</table>

### Screen numbers: Diagnostics operating area

<table>
<thead>
<tr>
<th>Screen</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm list</td>
<td>500</td>
</tr>
<tr>
<td>Messages</td>
<td>501</td>
</tr>
<tr>
<td>Alarm log</td>
<td>502</td>
</tr>
<tr>
<td>NC/PLC variable</td>
<td>503</td>
</tr>
</tbody>
</table>

### 14.6 Function interface

#### 14.6.1 Read/write NC variables

#### 14.6.1.1 User interface

The PLC user program can read or write a maximum of eight NC variables simultaneously via the NC/PLC interface "Read/write NC variable".
The following steps must be performed as part of a job (read/write):
1. Job specification (Page 1101)
2. Job management: Start job (Page 1102)
3. Job management: Waiting for end of job (Page 1103)
4. Job management: Job completion (Page 1103)
5. Job evaluation (Page 1105)

Flow diagram of a job: See "Job management: Flow diagram (Page 1104)"

### 14.6.1.2 Job specification

#### Variable-specific job interface

Each variable that is to be processed in a job, must be specified in the variable-specific job
interface via its parameters. The general identifiers are discussed in more detail later for each
variable that can be accessed from the interface.

<table>
<thead>
<tr>
<th>DB120x 1)</th>
<th>Reading/writing NC data (PLC → NCK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>Bit 7</td>
</tr>
<tr>
<td>DBB1000</td>
<td></td>
</tr>
<tr>
<td>DBB1001</td>
<td></td>
</tr>
<tr>
<td>DBW1002</td>
<td></td>
</tr>
<tr>
<td>DBW1004</td>
<td></td>
</tr>
<tr>
<td>DBW1006</td>
<td></td>
</tr>
<tr>
<td>DBD1008</td>
<td>Write: Data to NC variable x (data type of the variables: 1...4 bytes)(^2)</td>
</tr>
<tr>
<td>DBD1012</td>
<td></td>
</tr>
<tr>
<td>DBD1016</td>
<td>Write: Data to NC variable x (REAL) (^3)</td>
</tr>
<tr>
<td>DBD1020</td>
<td>Write: Data to NC variable x (DWORD/DINT) (^3)</td>
</tr>
<tr>
<td>DBW1024</td>
<td>Write: Data to NC variable x (WORD/INT) (^3)</td>
</tr>
<tr>
<td>DBB1026</td>
<td>Write: Data to NC variable x (BYTE) (^3)</td>
</tr>
<tr>
<td>DBB1027</td>
<td></td>
</tr>
</tbody>
</table>

1) \(\text{DB120}x\), with \(x = 0 \ldots 7\) corresponds to variable 1 ... 8.
2) Only for predefined variables of the "Read/write NC variable" user interface
3) Only for variables from DB9910 NC_DATA
Note

Channel-specific variables

When reading/writing channel-specific variables, only the variables of exactly one channel may be addressed in a job.

Drive-specific variables

When reading/writing drive-specific variables, only the variables of exactly one SERVO drive object may be addressed in a job. The SERVO drive object must be assigned to a machine axis of the NC. The line index corresponds to the logical drive number.

Error case

In the event of an error, reading/writing variables from different drive objects, or simultaneously from a channel and a drive object, an error message is output:

DB1200.DBX3000.1 == 1 (error occurred)

Example: Reading a variable of the "Location type" as the fourth variable

DB1203.DB1000: 7
DB1203.DB1001: -
DB1203.DBW1002: <Location number>
DB1203.DBW1004: <Magazine number>
DB1203.DBW1006: -
DB1203.DBD1008: -

Example: Writing a variable as the fourth variable

To write a data item to the NC, the value must be entered into the double word DBD1008:

DB1203.DB1000: <Variable index>
DB1203.DB1001: <Area number>
DB1203.DBW1002: <Column index>
DB1203.DBW1004: <Line index>
DB1203.DBW1006: -
DB1203.DBD1008: <value>

14.6.1.3 Job management: Start job

The following data must be written by the user to the global job interface:

<table>
<thead>
<tr>
<th>DB120x</th>
<th>Reading/writing NC data (PLC → NCK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>Bit 7</td>
</tr>
<tr>
<td>DBB 0</td>
<td></td>
</tr>
<tr>
<td>DBB 1</td>
<td></td>
</tr>
</tbody>
</table>

1) DB120x, with x = 0 ... 7 corresponds to variable 1 ... 8.
Job type
- Read variable: DB1200.DBX0.1 = 0
- Write variable: DB1200.DBX0.1 = 1

Start job
The start signal must be set to start the job via a specified number of variables:
DB1200.DBX0.0 = 1

Note
A new job can only be started if the previous job was completed. See Section "Job management: Waiting for end of job (Page 1103)". The execution of a job may take several PLC cycles and vary depending on the utilization. Therefore, the time for this function cannot be defined.

14.6.1.4 Job management: Waiting for end of job
The end of the job is always signaled back by the NC for the whole job in the global event interface. The signals can only be read by the PLC user.

### DB120x Reading/writing NC data (NCK → PLC)

<table>
<thead>
<tr>
<th>Byte</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBB 2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Error in job</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Job completed</td>
<td></td>
</tr>
</tbody>
</table>

1) DB120x, with x = 0 ... 7 corresponds to variable 1 ... 8.

Job status
- End of job without error
  DB1200.DBX2000.0 == 1 AND DB1200.DBX2000.1 == 0
- End of job with error
  DB1200.DBX2000.0 == 1 AND DB1200.DBX2000.1 == 1

Possible error causes
- Number of variables (DB1200.DBB1) out of the valid range
- Variable index (DB1200.DBB1000) out of the valid range
- Simultaneous reading/writing of NC data from different servo drive objects

14.6.1.5 Job management: Job completion
Requirement
In order to complete the job, the start signal of the job must be reset from the PLC user program after detection of the end of the job:
DB1200.DBX0.0 = 0
Feedback signal
As feedback, the NC resets the status signals:

- DB1200.DBX2000.0 == 0
- DB1200.DBX2000.1 == 0

The job is now completed.

14.6.1.6 Job management: Flow diagram

1. Start job:
   DB1200.DBX0.0 (start) = 1

2. Waiting for end of job:
   DB1200.DBX2000.0 (job completed) == 1 AND
   DB1200.DBX2000.1 (error in job) == 0
   ⇒ Reset job request:
   DB1200.DBX0.0 = 0 (start)

3. With DB1200.DBX0.0 = 0 (start), the job is completed by the basic PLC program:
   DB1200.DBX2000.0 (job completed) = 0

4. Waiting for end of job:
   DB1200.DBX2000.0 (job completed) == 0 AND
   DB1200.DBX2000.1 (error in job) == 1
   ⇒ Perform error handling
   ⇒ Reset job request:
   DB1200.DBX0.0 (start) = 0

5. With DB1200.DBX0.0 == 0 (start), the job is completed by the basic PLC program:
   DB1200.DBX2000.1 (error in job) = 0

6. If DB1200.DBX0.0 (start) is reset before the end of job is signaled by the basic PLC program, the job is executed without further feedback.
14.6.1.7 Job evaluation

Key statement

The variable-specific result interface must be evaluated for each variable processed in the job.

<table>
<thead>
<tr>
<th>DB120x</th>
<th>NC services (NC → PLC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBB3000</td>
<td>Bit 7 Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 Bit 0</td>
</tr>
<tr>
<td></td>
<td>Error has occurred Variable valid</td>
</tr>
<tr>
<td>DBB3001</td>
<td>Access result (see &quot;Access result&quot; below)</td>
</tr>
<tr>
<td>DBW3002</td>
<td>---</td>
</tr>
<tr>
<td>DBD3004</td>
<td>Read: Data from NC variable x (data type of the variables: 1...4 bytes)</td>
</tr>
<tr>
<td>DBD3008</td>
<td>---</td>
</tr>
<tr>
<td>DBD3012</td>
<td>---</td>
</tr>
<tr>
<td>DBD3016</td>
<td>Read: Data from NC variable x (REAL)</td>
</tr>
<tr>
<td>DBD3020</td>
<td>Read: Data from NC variable x (DWORD / DINT)</td>
</tr>
<tr>
<td>DBW3024</td>
<td>Read: Data from NCK variable x (WORD / INT)</td>
</tr>
<tr>
<td>DBB3026</td>
<td>Read: Data from NCK variable x (BYTE)</td>
</tr>
<tr>
<td>DBB3027</td>
<td>--- --- --- --- --- --- ---</td>
</tr>
</tbody>
</table>

1) DB120x, with x = 0 ... 7 corresponds to variable 1 ... 8.
2) Only for predefined variables of the "Read/write NC variable" user interface
3) Only for variables from DB9910 NC_DATA

Note

Channel-specific variables

When reading/writing channel-specific variables, only the variables of exactly one channel may be addressed in a job.

Drive-specific variables

When reading/writing drive-specific variables, only the variables of exactly one SERVO drive object may be addressed in a job. The SERVO drive object must be assigned to a machine axis of the NC. The line index corresponds to the logical drive number.

Error case

In the event of an error, reading/writing variables from different drive objects, or simultaneously from a channel and a drive object, an error message is output:

DB1200.DBX3000.1 == 1 (error occurred)
Access result

NC variables

<table>
<thead>
<tr>
<th>DBB3001</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>3</td>
<td>Access to object is not permitted</td>
</tr>
<tr>
<td>5</td>
<td>Invalid address</td>
</tr>
<tr>
<td>10</td>
<td>Object does not exist</td>
</tr>
</tbody>
</table>

Drive-specific variables

If an error occurs while reading/writing a drive-specific variable (DB1200.DBX3000.1 == 1), an error number is displayed in the access result which is based on the error numbers defined in the PROFIdrive profile.

<table>
<thead>
<tr>
<th>DBB3001</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>&lt;error number of the PROFIdrive profile&gt; + 20_H or 36_D</td>
</tr>
</tbody>
</table>

Determining the meaning of the access result:

1. Calculate the error number of the PROFIdrive profile
   
   \[ \text{<error number of the PROFIdrive profile> = access result - 20_H or 36_D} \]

2. Determine the meaning of the error number of the PROFIdrive profile

The error numbers of the PROFIdrive profile are described in:

References
Function Manual, SINAMICS S120 Drive Functions; Section "Communication" > "Communication according to PROFIdrive" > "Acyclic communication" > "Structure of the requests and responses" > Subsection "Error values in parameter responses"

Examples: Job status

Job without error

- DB1200.DBX3000.0 == 1 (variable valid) **AND**
- DB1200.DBX3000.1 == 0 (no error occurred)

Result:
- DB1200.DBB3001 == 0 (access result: "No error")
- DB1200.DBD3004 == <read value>

Job with error

- DB1200.DBX3000.0 == 0 (variable not valid) **AND**
- DB1200.DBX3000.1 == 1 (error occurred)

Result:
- DB1200.DBB3001: For possible error causes, see "Access result" above
14.6.1.8 Operable variables

The following variables are available:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>cuttEdgeParam</td>
<td>Compensation value parameters and cutting edge list with D numbers for a tool</td>
</tr>
<tr>
<td>numCuttEdgeParams</td>
<td>Number of P elements of a cutting edge</td>
</tr>
<tr>
<td>linShift</td>
<td>Translation of a settable zero-point offset</td>
</tr>
<tr>
<td>numMachAxes</td>
<td>Number of the highest existing channel axis</td>
</tr>
<tr>
<td>rpa</td>
<td>R parameters</td>
</tr>
<tr>
<td>actLineNumber</td>
<td>Line number of the current NC block</td>
</tr>
<tr>
<td>$TC_MPPx</td>
<td>Magazine location data</td>
</tr>
<tr>
<td>r0078[1]</td>
<td>Current actual value, torque-generating</td>
</tr>
<tr>
<td>r0079[1]</td>
<td>Torque setpoint at the output of the speed controller</td>
</tr>
<tr>
<td>r0081</td>
<td>Torque utilization in percent</td>
</tr>
<tr>
<td>r0082[1]</td>
<td>Active power actual value</td>
</tr>
<tr>
<td>TEMP_COMP_x</td>
<td>Temperature compensation data</td>
</tr>
</tbody>
</table>

Variable "cuttEdgeParam"

Compensation value parameters and cutting edge list with D numbers for a tool

The meanings of the individual parameters depend on the type of the tool in question. Currently, 25 parameters are reserved for each tool edge (but only a part of them is loaded with values). To be able to remain flexible for future extensions, it is not recommended to use a fixed value of 25 parameters for calculation, but the variable value 'numCuttEdgeParams' (variable index 2).

For a detailed description of tool parameters, see Section W1: Tool offset (Page 1363).

<table>
<thead>
<tr>
<th>Address</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB120x.DBB1000</td>
<td>1</td>
</tr>
<tr>
<td>DB120x.DBB1001</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW1002</td>
<td>(Cutting edge No. - 1) * numCuttEdgeParams + ParameterNr (WORD)</td>
</tr>
<tr>
<td>DB120x.DBW1004</td>
<td>T number (1...32000) (WORD)</td>
</tr>
<tr>
<td>DB120x.DBD1008</td>
<td>Writing: Data to NCK variable x (data type: REAL)</td>
</tr>
<tr>
<td>DB120x.DBW3004</td>
<td>Reading: Data from NCK variable x (data type: REAL)</td>
</tr>
</tbody>
</table>

Variable "numCuttEdgeParams"

Number of P elements of a cutting edge

<table>
<thead>
<tr>
<th>Address</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB120x.DBB1000</td>
<td>2</td>
</tr>
<tr>
<td>DB120x.DBB1001</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW1002</td>
<td>-</td>
</tr>
</tbody>
</table>
### Variable "linShift"

**Translation of a settable work offset (channel-specific settable frames)**

The variable only exists if MD18601 MM_NUM_GLOBAL_USER_FRAMES > 0.

The following frame indices are available:

<table>
<thead>
<tr>
<th>Index</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ACTFRAME = actual resulting work offset</td>
</tr>
<tr>
<td>1</td>
<td>IFRAME = actual settable work offset</td>
</tr>
<tr>
<td>2</td>
<td>PFRAME = actual programmable work offset</td>
</tr>
<tr>
<td>3</td>
<td>EXTRAME = actual external work offset</td>
</tr>
<tr>
<td>4</td>
<td>TOTFRAME = actual total work offset = total of ACTFRAME and EXTRAME</td>
</tr>
<tr>
<td>5</td>
<td>ACTBFRAME = actual total base frame</td>
</tr>
<tr>
<td>6</td>
<td>SETFRAME = actual 1st system frame (PRESET, scratching)</td>
</tr>
<tr>
<td>7</td>
<td>EXTSFRAME = actual 2nd system frame (PRESET, scratching)</td>
</tr>
<tr>
<td>8</td>
<td>PARTFRAME = actual 3rd system frame (TCARR and PAROT with orientable tool carrier)</td>
</tr>
<tr>
<td>9</td>
<td>TOOLFRAME = actual 4th system frame (TOROT and TOFRAME)</td>
</tr>
<tr>
<td>10</td>
<td>MEASFRAME = result frame for workpiece and tool gauging</td>
</tr>
<tr>
<td>11</td>
<td>WPFRAME = actual 5th system frame (workpiece reference points)</td>
</tr>
<tr>
<td>12</td>
<td>CYCFRAME = actual 6th system frame (cycles)</td>
</tr>
</tbody>
</table>

### Variable "numMachAxes"

**Number of the highest existing channel axis**

If there are no channel axis gaps, the value of the variables is also the number of available axes in the channel.
Variable "rpa"

R parameters

<table>
<thead>
<tr>
<th>Address</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB120x.DBB1000</td>
<td>5</td>
</tr>
<tr>
<td>DB120x.DBB1001</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW1002</td>
<td>R number + 1</td>
</tr>
<tr>
<td>DB120x.DBW1004</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBD1008</td>
<td>Writing: Data to NCK variable x (data type: REAL)</td>
</tr>
<tr>
<td>DB120x.DBW3004</td>
<td>Reading: Data from NCK variable x (data type: REAL)</td>
</tr>
</tbody>
</table>

Variable "actLineNumber"

Line number of the current NC block

<table>
<thead>
<tr>
<th>No.</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>≥ 1</td>
<td>Line number of the current NC block</td>
</tr>
<tr>
<td>0</td>
<td>No line number available because the program has not started</td>
</tr>
<tr>
<td>-1</td>
<td>No line number available: Error</td>
</tr>
<tr>
<td>-2</td>
<td>No line number available: Suppression of block display with DISPLOF is active</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB120x.DBB1000</td>
<td>6</td>
</tr>
<tr>
<td>DB120x.DBB1001</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW1002</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW1004</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBD1008</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW3004</td>
<td>Reading: Data from NCK variable x (data type: INT)</td>
</tr>
</tbody>
</table>

Tool management: Magazine location data

Location type ($TC_MPP2)

<table>
<thead>
<tr>
<th>Address</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB120x.DBB1000</td>
<td>7</td>
</tr>
<tr>
<td>DB120x.DBB1001</td>
<td>-</td>
</tr>
</tbody>
</table>
### Address

<table>
<thead>
<tr>
<th>Address</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB120x.DBW1002</td>
<td>Location number (1 … 31999)</td>
</tr>
<tr>
<td>DB120x.DBW1004</td>
<td>Magazine number (1 … 9999)</td>
</tr>
<tr>
<td>DB120x.DBD1008</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW3004</td>
<td><strong>Reading</strong>: Value of the NCK variable (data type: WORD)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>Location type for virtual location</td>
</tr>
<tr>
<td>0</td>
<td>&quot;match all&quot; (buffer)</td>
</tr>
<tr>
<td>9999</td>
<td>undefined (no virtual location)</td>
</tr>
</tbody>
</table>

### Location status ($TC_MPP4)

<table>
<thead>
<tr>
<th>Address</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB120x.DBB1000</td>
<td>8</td>
</tr>
<tr>
<td>DB120x.DBB1001</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW1002</td>
<td>Location number (1 … 31999)</td>
</tr>
<tr>
<td>DB120x.DBW1004</td>
<td>Magazine number (1 … 9999)</td>
</tr>
<tr>
<td>DB120x.DBD1008</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW3004</td>
<td><strong>Reading</strong>: Value of the NCK variable (data type: WORD)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Blocked</td>
</tr>
<tr>
<td>2</td>
<td>free (&lt;&gt; occupied)</td>
</tr>
<tr>
<td>4</td>
<td>reserved for tool in buffer</td>
</tr>
<tr>
<td>8</td>
<td>reserved for tool to be loaded</td>
</tr>
<tr>
<td>16</td>
<td>occupied in left half location</td>
</tr>
<tr>
<td>32</td>
<td>occupied in right half location</td>
</tr>
<tr>
<td>64</td>
<td>occupied in upper half location</td>
</tr>
<tr>
<td>128</td>
<td>occupied in lower half location</td>
</tr>
</tbody>
</table>

### T No. of tool at this location ($TC_MPP6)

<table>
<thead>
<tr>
<th>Address</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB120x.DBB1000</td>
<td>9</td>
</tr>
<tr>
<td>DB120x.DBB1001</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW1002</td>
<td>Location number (1 … 31999)</td>
</tr>
<tr>
<td>DB120x.DBW1004</td>
<td>Magazine number (1 … 9999)</td>
</tr>
<tr>
<td>DB120x.DBD1008</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW3004</td>
<td><strong>Reading</strong>: T number of the tool at this location (data type: WORD)</td>
</tr>
</tbody>
</table>

### Variable r0078[1]

- Drive object: SERVO, SERVO_AC, SERVO_I_AC
- CO: Current actual value, torque-generating \([A_{\text{rms}}]\)
- Index [1]: smoothed with p0045
Variable r0079[1]

- Drive object: SERVO, SERVO_AC, SERVO_I_AC
- CO: Torque setpoint at the output of the speed controller (before clock cycle interpolation) [Nm]
- Index [1]: smoothed with p0045

Variable r0081

- Drive object: SERVO, SERVO_AC, SERVO_I_AC
- CO: Torque utilization in percent

Variable r0082[1]

- Drive object: SERVO, SERVO_AC, SERVO_I_AC
- CO: Active power actual value [kW]
- Index [1]: smoothed with p0045
Temperature compensation

Variable "TEMP_COMP_ABS_VALUE" (SD43900)
Position-independent temperature compensation value

<table>
<thead>
<tr>
<th>Address</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB120x.DBB1000</td>
<td>14</td>
</tr>
<tr>
<td>DB120x.DBB1001</td>
<td>No. of the axis (1, 2, ...)</td>
</tr>
<tr>
<td>DB120x.DBW1002</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW1004</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBD1008</td>
<td>Writing: Data to NCK variable x (data type: REAL)</td>
</tr>
<tr>
<td>DB120x.DBW3004</td>
<td>Reading: Data from NCK variable x (data type: REAL)</td>
</tr>
</tbody>
</table>

Variable "TEMP_COMP_SLOPE" (SD43910)
Gradient for position-dependent temperature compensation

<table>
<thead>
<tr>
<th>Address</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB120x.DBB1000</td>
<td>15</td>
</tr>
<tr>
<td>DB120x.DBB1001</td>
<td>No. of the axis (1, 2, ...)</td>
</tr>
<tr>
<td>DB120x.DBW1002</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW1004</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBD1008</td>
<td>Writing: Data to NCK variable x (data type: REAL)</td>
</tr>
<tr>
<td>DB120x.DBW3004</td>
<td>Reading: Data from NCK variable x (data type: REAL)</td>
</tr>
</tbody>
</table>

Variable "TEMP_COMP_REF_POSITION" (SD43920)
Reference position for position-dependent temperature compensation

<table>
<thead>
<tr>
<th>Address</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB120x.DBB1000</td>
<td>16</td>
</tr>
<tr>
<td>DB120x.DBB1001</td>
<td>No. of the axis (1, 2, ...)</td>
</tr>
<tr>
<td>DB120x.DBW1002</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW1004</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBD1008</td>
<td>Writing: Data to NCK variable x (data type: REAL)</td>
</tr>
<tr>
<td>DB120x.DBW3004</td>
<td>Reading: Data from NCK variable x (data type: REAL)</td>
</tr>
</tbody>
</table>
Variable "TOOL_TEMP_COMP" (SD42960[...])
Temperature compensation referred to the tool

<table>
<thead>
<tr>
<th>Address</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB120x.DBB1000</td>
<td>17</td>
</tr>
<tr>
<td>DB120x.DBB1001</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBW1002</td>
<td>Index + 1 (1, 2, 3)</td>
</tr>
<tr>
<td>DB120x.DBW1004</td>
<td>-</td>
</tr>
<tr>
<td>DB120x.DBD1008</td>
<td>Writing: Data to NCK variable x (data type: REAL)</td>
</tr>
<tr>
<td>DB120x.DBW3004</td>
<td>Reading: Data from NCK variable x (data type: REAL)</td>
</tr>
</tbody>
</table>

14.6.1.9 Specifying selected NC variables

The selected NC variables are specified via the DB9910 data block (selected NC variable). The length of the data block depends on the number of NC variables selected in the variables list. The variables list can contain maximum 42 selected NC variables. The DB9910 data block contains the data for variable addressing and data type conversion for each NC variable.

<table>
<thead>
<tr>
<th>DB9910 (read selected NC variable)</th>
<th>(Interface: PLC → NC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBB 0</td>
<td>Variable index Variable 1</td>
</tr>
<tr>
<td>DBB 1</td>
<td>Syntax ID</td>
</tr>
<tr>
<td>DBB 2</td>
<td>Area</td>
</tr>
<tr>
<td>DBB 3</td>
<td>Unit</td>
</tr>
<tr>
<td>DBW 4</td>
<td>Column index</td>
</tr>
<tr>
<td>DBW 6</td>
<td>Line index</td>
</tr>
<tr>
<td>DBB 8</td>
<td>Block</td>
</tr>
<tr>
<td>DBB 9</td>
<td>Number of lines</td>
</tr>
<tr>
<td>DBB 10</td>
<td>Type</td>
</tr>
<tr>
<td>DBB 11</td>
<td>Length</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>DBB 492</td>
<td>Variable index Variable 42</td>
</tr>
<tr>
<td>DBB 493</td>
<td>Syntax ID</td>
</tr>
<tr>
<td>DBB 494</td>
<td>Area</td>
</tr>
<tr>
<td>DBB 495</td>
<td>Unit</td>
</tr>
<tr>
<td>DBW 496</td>
<td>Column index</td>
</tr>
<tr>
<td>DBW 498</td>
<td>Line index</td>
</tr>
<tr>
<td>DBB 500</td>
<td>Block</td>
</tr>
<tr>
<td>DBB 501</td>
<td>Number of lines</td>
</tr>
<tr>
<td>DBB 502</td>
<td>Type</td>
</tr>
<tr>
<td>DBB 503</td>
<td>Length</td>
</tr>
</tbody>
</table>
Variable index

The variable index refers to the name of the NC variable. The variable index consists of a start value 100 and the offset of the NC variable in the list (0 to 41).

The variable index is entered as DBB1000 A_VarIdx in the DB120x RW_NCDx user interface.

The comment contains the following data record with space as separator:

- Area
- Block
- VariablenName
- VarType
- Column
- VarAnzByte

Extended user interface

The data test in data blocks means the value of a variable can be written to an address in the data block that has the same type, e.g. a REAL value can be written only to a REAL address (e.g. with MOV_R). The "Read/write NC variable" user interface contains currently only one address of the DWORD type for the value to be written (DBD1008). This means a REAL value can be written only via a temporary variable, flag or accumulator. The same is also true for reading (DBD3004). For this reason, the "Read/write NC variable" user interface is extended. For reading and writing one address for each type is added: REAL, DWORD/DINT, WORD/INT, BYTE and BOOL (DBD1016 … DBD1027 or DBD3016 … DBD3027). These new addresses are used by the PLC firmware only for those variables selected with the NC variables editor – and entered into DB9910 NC_DATA when compiled (variable index ≥ 100). The NC variables currently defined in the "Read/write NC variable" user interface continue to use the old addresses (DBD1008 or DBD3004).

14.6.2 Program instance services (PI services)

14.6.2.1 User interface

Job specification

PI services are specified via the job interface (DB1200 from offset 4000).

<table>
<thead>
<tr>
<th>Byte</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBB 4000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Start 1)</td>
</tr>
<tr>
<td>DBB 4001</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>PI index 2)</td>
</tr>
<tr>
<td>DBB 4002</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>DBB 4003</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

1) "Start" indicates the start of the PI service.
2) "PI index" indicates the index of the PI service.
### Overview of the PI services:

- **PI service ASUB** (Page 1117)
- **PI service LOGOUT** (Page 1118)
- **PI service DATA_SAVE** (Page 1118)
- **PI service TMMVTL** (Page 1119)

### Job feedback

The PLC provides feedback as to whether the started PI service was successful or not successful in the **result interface** (DB1200 from offset 5000).

The job end is signaled using the following signals:

- DB1200.DBX5000.0
- DB1200.DBX5000.1

The signals are written by the PLC; therefore, they can only be read by the user.

A job has been completed as soon as the user resets the "Start" signal (DB1200.DBX4000.1). Status signals DB1200.DBX5000.0 and .1 are then set to zero.

### Basic Functions

Function Manual, 01/2015, 6FC5397-0BP40-5BA2
Possible error causes

- The index of the parameterized PI service (DB1200.DBB4001) is outside the valid range
- Parameter error

PI services: Cycle diagram

1. User sets the signal "Start", job execution starts.
2. After the PLC firmware signals "Job completed", the user resets the signal "Start".
3. By resetting the signal "Start", the PLC firmware resets the signal "Job completed".
4. After the PLC firmware signals "Error in job", the user resets the signal "Start".
5. By resetting the signal "Start", the PLC firmware resets the signal "Error in job".
6. If the user accidentally resets the signal "Start" before one of the signals "Job completed" or "Error in job" is received, then the result signals for this job are not updated. However, the job is executed.

14.6.2.2 PI services

PI service ASUB

Function

With the "ASUB" PI service, it is possible to assign one program each to the interrupt numbers 1 and 2 from the PLC. These programs must be present in the NC in the machine manufacturer directory (CMA) with fixed program names:

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Directory</th>
<th>Program name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>_N_CMA_DIR</td>
<td>PLCASUP1_SPF</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>PLCASUP2_SPF</td>
</tr>
</tbody>
</table>

If the programs are not available, then they must be created. A power on reset must then be performed on the NC.

The PI service "ASUB" must only be executed once after the control powers up. The assignment of the interrupt to the program is retained until the control powers up again.
Job specification

**PI service: ASUB**

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB1200.DBW4001</td>
<td>PI index</td>
<td>1) 1 (interrupt 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2) 2 (interrupt 2)</td>
</tr>
<tr>
<td>DB1200.DBW4004</td>
<td>Parameter 1: LIFTFAST</td>
<td>0 (FALSE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (TRUE)</td>
</tr>
<tr>
<td>DB1200.DBW4006</td>
<td>Parameter 2: BLSYNC</td>
<td>0 (FALSE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (TRUE)</td>
</tr>
</tbody>
</table>

1) PI index = 1 ⇒ assignment: Interrupt 1 for _N_CMA_DIR/PLCASUP1_SPF, interrupt priority = 1
   PI index = 2 ⇒ assignment: Interrupt 2 for _N_CMA_DIR/PLCASUP2_SPF, interrupt priority = 2
2) After the interrupt has been initiated, LIFTFAST first results in fast retraction (fast lift). Only then is the interrupt routine executed.

**Reference**

Function Manual, Special Functions
Chapter "8 R3: Extended stop and retract" > "8.2 control-managed ESR - only 840D sl" > "Retract"
3) BLSYNC ensures that after the interrupt has been initiated, the system first waits until the actual block has been executed. Only then is the interrupt routine executed.

**Supplementary conditions**

- The PI service "ASUB" may only be executed when the channel in which it is requested is in the RESET state.
- If, for an event-controlled program call (Prog Event), "power up" is configured as initiating event (MD20108 $MC_PROG_EVENT_MASK), then the PI service "ASUB" may only be started after the ProgEvent program has been completed (PROG_EVENT_SPF or MD11620 $MN_PROG_EVENT_NAME=<User_Prog_event_SPF>).

**See also**

Job specification (Page 1114)
Job feedback (Page 1115)

**PI service LOGOUT**

**Function**

The password last transferred to the NC is reset.

**Job specification**

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB1200.DBW4001</td>
<td>PI index</td>
<td>3 (reset password)</td>
</tr>
</tbody>
</table>
PI service DATA_SAVE

Function

Perform data saving.

Job specification

<table>
<thead>
<tr>
<th>PI service: DATA_SAVE</th>
<th>Description</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB1200.DBW4001</td>
<td>PI index</td>
<td>4</td>
</tr>
</tbody>
</table>

Perform data saving of the current NC state on the system CompactFlash card.

Note

The saved state of the NC can be loaded during the next control power-up via "Reload saved user data" in the "Startup menu".

References

SINUMERIK 828D Commissioning Manual; Section "CNC commissioning" > "Scope of delivery and requirements" > "Control run-up"

PI service TMMVTL

Function

Using the PI service TMMVTL, it is possible to request a job from the PLC to relocate a tool. As a result of the PI service, the tool manager carries out an empty location search in the target magazine for the tool specified in the PI service (tool number or source location number/source magazine number) The PLC then receives a job to relocate the tool via DB41xx.DBB0.

Job specification

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB1200.DBW4001</td>
<td>PI index</td>
<td>5</td>
</tr>
<tr>
<td>DB1200.DBW4004</td>
<td>Parameter 1: Tool number ¹)</td>
<td>-1, 1 ... 31999</td>
</tr>
<tr>
<td>DB1200.DBW4006</td>
<td>Parameter 2: Source location number ¹)</td>
<td>-1, 1 ... 31999</td>
</tr>
<tr>
<td>DB1200.DBW4008</td>
<td>Parameter 3: Source magazine number ¹)</td>
<td>-1, 1 ... 9999</td>
</tr>
<tr>
<td>DB1200.DBW4010</td>
<td>Parameter 4: Target location number ²)</td>
<td>-1, 1 ... 31999</td>
</tr>
</tbody>
</table>
14.6 Function interface

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB1200.DBW4012</td>
<td>Parameter 5: Target magazine number 3)</td>
<td>-1, 1 … 9999</td>
</tr>
</tbody>
</table>

1) The tool can be optionally specified using:
   - Tool number (T number)
   - Source location and source magazine number

For the unused parameters of the other version, a value of -1 must be entered.

2) With the target location number = -1, a search is made in the complete magazine for an empty location for the tool according to the search strategy that has been selected. If a target location is specified, then a check is made as to whether the location with the specified target location number is free and suitable for the particular tool.

3) For a target magazine number = -1, a search is made in a buffer for the tool corresponding to the assignment obtained from $TC_MDP2.

Application examples

- When using buffers to return the tool (for example Toolboy and/or shifter), an explicit empty location search in the magazine may be needed during the asynchronous return transport. In this case, the PLC does not have to note the original location, the PI service TMMVT0 searches for a suitable location.

- A tool is to be moved from a background magazine to the front magazine.

14.6.3 PLC user alarms

14.6.3.1 User interface

Note

Although the name user “alarms” is used in the following, it is only defined as to whether it involves a message or an alarm when entering the particular cancel criterion in bits 6 and 7 of machine data MD14516[x].

The user alarms are created in the HMI and are prepared for automatic processing. The user interface of the DB1600 permits:

- these 248 user alarms of numbers 700000 to 700247 to be activated,
- to be provided with an additional numerical parameter, and
- to be activated and acknowledged as well as
- to evaluate the system reactions initiated by it.

Activation interface of the user alarms

Each user alarm is activated using its assigned activation bit. These bits are set in the activation interface (DB1600 from offset 0):
A new user alarm is activated with a 0/1 edge of the particular bit.

### Variables interface of the user alarms

Each user alarm can be given a variable as parameter. One double word each is reserved for this purpose in the variable interface from offset 1000. As a consequence, valid offsets must be divisible by 4.
Alarm response and cancel criterion

These two terms involve the conception and configuration of user alarms. The following attributes can be specified for each alarm:

- **Alarm response**: How the controller responds when an error occurs.
- **Cancel criterion**: What must be done to cancel the alarm again or acknowledge it. The cancel criterion simultaneously defines the alarm priority.

The setpoints are configured in the following machine data and their actual values are signaled together in one byte of the user interface: The setpoints are configured bit-coded in: MD14516[x], 0 ≤ x ≤ 247; whereby x = user alarm number - 700000

Active alarm response and cancel criterion

The present **active alarm responses** (i.e. the actual responses) and the **active cancel criteria** can be globally read out from the interface:

<table>
<thead>
<tr>
<th>DB1600</th>
<th>Active alarm response [r]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>Bit 7</td>
</tr>
<tr>
<td>DBB 2000</td>
<td>POWER ON</td>
</tr>
<tr>
<td>DBB 2001</td>
<td></td>
</tr>
<tr>
<td>DBB 2002</td>
<td></td>
</tr>
<tr>
<td>DBB 2003</td>
<td></td>
</tr>
</tbody>
</table>

One bit is set if for at least one active alarm the corresponding response or the corresponding cancel criterion is configured. It is canceled, if this response/cancel criterion is no longer configured for any of the alarms present.

The codings of the cancel criteria and the corresponding priorities are:

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>User alarm is acknowledged by</th>
<th>Type</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Resetting of the activation bit</td>
<td>Message</td>
<td>Low</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Acknowledgement in DB1600.DBX3000.0 (refer below)</td>
<td>Alarm</td>
<td>Medium</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Power on</td>
<td>Alarm</td>
<td>High</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Reserved (internally evaluated as {1, 0})</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note**

If none of bits 0 to 4 are set for an alarm/message in the machine data, then this defines that it involves a so-called "display message" that has no effect on the system. This especially indicates that also bits 6 and 7 of the corresponding machine data are evaluated.
Active alarm responses and cancel criteria of the user alarms

The present active alarm responses (i.e. the actual responses) and the active cancel criteria can be globally read-out of the interface.

<table>
<thead>
<tr>
<th>DB1600</th>
<th>Active alarm response [r]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data block</td>
<td>Bit block</td>
</tr>
<tr>
<td>Byte</td>
<td>Bit 7</td>
</tr>
<tr>
<td>DBB 2000</td>
<td>POWER ON</td>
</tr>
<tr>
<td>DBB 2001</td>
<td></td>
</tr>
<tr>
<td>DBB 2002</td>
<td></td>
</tr>
<tr>
<td>DBB 2003</td>
<td></td>
</tr>
</tbody>
</table>

One bit is set if for at least one active alarm the corresponding response or the corresponding cancel criterion is configured. It is canceled, if this response/cancel criterion is no longer configured for any of the alarms present.

Acknowledgement interface of the user alarms

Requirement

Requirement to acknowledge a user alarm is that the corresponding activation bit is reset.

- Messages with cancel criterion {0,0} then disappear automatically from the display.
- Alarms with cancel criterion {0,1} are canceled by the acknowledgement bit Ack.
- Alarms with cancel criterion {1,0} are not influenced when the acknowledgement bit is set and can only be canceled by a Power On.

<table>
<thead>
<tr>
<th>DB1600</th>
<th>Alarm acknowledgement [r/w]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>Bit 7</td>
</tr>
<tr>
<td>DBB3000</td>
<td>Ack</td>
</tr>
<tr>
<td>DBB3001</td>
<td></td>
</tr>
<tr>
<td>DBB3002</td>
<td></td>
</tr>
<tr>
<td>DBB3003</td>
<td></td>
</tr>
</tbody>
</table>

Interface to HMI.

The PLC can transfer eight messages or alarms for display on the HMI, which are displayed in the sequence that they occur.
When additional messages/alarms occur, the first seven are kept in the HMI, and the latest message or the latest alarm is displaced from one that has just occurred according to the following rules:

- System message/alarm displaces user message/alarm
- Messages/alarms with a higher priority displace those of a lower priority.

The first seven messages/alarms are kept in the display because it is very probable that these define the cause of the problem and the following are just of a secondary nature. However, if one or several messages/alarms are acknowledged and therefore cleared, then a corresponding number of alarms/messages that have been received move up in the HMI.

14.6.4 PLC axis control

14.6.4.1 Overview

Target

The PLC can control eight axes or spindles via data blocks of the user interface; the axis/spindle is specified by its DB number:

- **DB380x** PLC → NCK interface
- **DB390x** NCK → PLC interface
- **DB390x** Axis index: 0 ≤ x ≤ 7, whereby axis index = axis number - 1

The following functions are supported:

- Positioning axes
- Spindle positioning
- Rotate spindle
- Oscillate spindle
- Indexing axes

References:

- Function Manual, Extended Functions; Positioning Axes (P2) and Indexing axes (T1)
- Function Manual, Basic Functions; Spindles (S1)

Precondition

The axis to be controlled must be assigned to the PLC. An axis can be interchanged between NC and PLC using the user interface "Axis interchange" (DB3800.DB8/DB3900.DB8).
Function start

Each function is activated by the positive edge of the corresponding "Start" signal. This signal must remain a logical "1" until the function has been positively or negatively acknowledged (e.g. using Position reached = "1" or Error = "1"). The signal "Positioning axis active" = "1" indicates that the function is active and that the output signals are valid.

Interrupt

It is not possible to interrupt the function by resetting the start signal, but only via other interface signals (using the axis-specific signal Delete distance to go/spindle reset, DB380x DBX2.2).

The axis interface returns axis status signals that may need to be evaluated (e.g. exact stop, travel command, → DB390x).

If the axis/spindle is being traversed via the NC program when the PLC axis control is called (travel command present), then the function is only started after this movement has ended. No error code is output in this situation.

Axis disable

With the axis disable set, an axis controlled via PLC axis control will not move. Only a simulated actual value is generated. (Behavior as with NC programming).

14.6.4.2 User interface: Preparing the NC axis as PLC axis

Firstly, the axis/spindle must be requested from the PLC. This is realized via the following user interface:

Common signals to axis/spindle (excerpt)

Request axis or spindle:

<table>
<thead>
<tr>
<th>DB3800 ... 3807</th>
<th>Signals to axis/spindle [r/w]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data block</strong></td>
<td><strong>PLC → NCK interface</strong></td>
</tr>
<tr>
<td><strong>Byte</strong></td>
<td><strong>Bit 7</strong></td>
</tr>
<tr>
<td>DBB 0008</td>
<td>Request PLC axis/ spindle</td>
</tr>
</tbody>
</table>

Every change to a request bit at the interface (DB380x.DBX0008.7 or bit DB380x.DBX0008.0) must be signaled to the NC using a 0→1 edge of the activation signal (bit DB380x.DBX0008.4). This activation signal should be reset again after one cycle.
General signals from axis/spindle (excerpt)

State interrogation is possible via the interface DB390x.DBB8. However, for simulation, the axis machine data MD30350 MA_SIMU_AX_VDI_OUTPUT must be set.

<table>
<thead>
<tr>
<th>DB3900 ... 3907</th>
<th>Signals from axis/spindle [r]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data block</td>
<td>NCK → PLC interface</td>
</tr>
<tr>
<td>Byte</td>
<td>Bit 7</td>
</tr>
<tr>
<td>DBB 0008</td>
<td>PLC axis/spindle</td>
</tr>
</tbody>
</table>

Request and relinquish PLC axis

<table>
<thead>
<tr>
<th>t</th>
<th>DB380x.DBB0008</th>
<th>DB390x.DBB0008</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 1 0 0 0 0 1</td>
<td>PLC receives Signal &quot;axis interchange possible&quot;, &quot;NC axis&quot; and possibly &quot;neutral axis/spindle&quot; (x).</td>
</tr>
<tr>
<td></td>
<td>1 0 0 1 0 0 0 0</td>
<td>1 0 0 0 0 0 1</td>
<td>PLC requests axis.</td>
</tr>
<tr>
<td></td>
<td>1 0 0 0 0 0 0 0</td>
<td>1 0 1 0 0 0 1</td>
<td>PLC resets the activation signal.</td>
</tr>
</tbody>
</table>

Axis is assigned to the PLC, can be used and is then relinquished to the NC:

|   | 0 0 0 1 0 0 0 1 | 1 0 0 0 0 0 1  | PLC returns axis to NC. |
|   | 0 0 0 0 0 0 0 1 | 0 0 1 0 0 0 1  | PLC resets activation signal; Status "NC axis" and "Axis interchange possible". |

Axis is assigned to the PLC, can be used and is then enabled:

|   | 0 0 0 1 0 0 0 0 | 0 1 0 0 0 0 1  | PLC enables axis |
|   | 0 0 0 0 0 0 0 0 | 0 1 1 0 0 0 1  | PLC resets activation signal; Status "neutral axis/spindle", "NC axis" and "axis interchange possible". |

14.6.4.3 User interface: Functionality

The two tables provide an overview of the available interface signals. The precise description of the signals and the explanation of what signals are relevant for the individual functions are explained in the following.
Signals to PLC axis

<table>
<thead>
<tr>
<th>DB3800...3807</th>
<th>Signals to PLC axis [r/w]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data block</strong></td>
<td><strong>PLC → NCK interface</strong></td>
</tr>
<tr>
<td><strong>Byte</strong></td>
<td><strong>Bit 7</strong></td>
</tr>
<tr>
<td>DBB3000</td>
<td>Start positioning axis</td>
</tr>
<tr>
<td>DBB3001</td>
<td>-</td>
</tr>
<tr>
<td>DBB3002</td>
<td>Automatic gear selection</td>
</tr>
<tr>
<td>DBB3003</td>
<td>Indexing position</td>
</tr>
<tr>
<td>DBD3004</td>
<td>Position (REAL, with indexing axis: DINT)</td>
</tr>
<tr>
<td>DBD3008</td>
<td>Feedrate velocity (REAL), if &lt; 0, the value is taken from machine data POS_AX_VELO</td>
</tr>
</tbody>
</table>

The bits of the distance conditions and the direction of rotation definition define the particular positioning or traversing mode, only one of the bits must be set:

<table>
<thead>
<tr>
<th><strong>Meaning</strong></th>
<th><strong>Distance condition to be set</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Positioning absolute</td>
<td>No mode bit set</td>
</tr>
<tr>
<td>Positioning incremental</td>
<td>DBB3002.0 = 1</td>
</tr>
<tr>
<td>Positioning shortest distance</td>
<td>DBB3002.1 = 1</td>
</tr>
<tr>
<td>Positioning absolute, positive approach direction</td>
<td>DBB3003.1 = 1</td>
</tr>
<tr>
<td>Positioning absolute, negative approach direction</td>
<td>DBB3003.0 = 1</td>
</tr>
<tr>
<td>Direction of rotation as for M4</td>
<td>DBB3002.5 = 1</td>
</tr>
</tbody>
</table>

The remaining bits are used to specify and start the particular function, these function bits as well as position and velocity are explained in more detail for the individual functions.

Signals from PLC axis

<table>
<thead>
<tr>
<th>DB3900...3907</th>
<th>Signals from PLC axis [r]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data block</strong></td>
<td><strong>PLC → NCK interface</strong></td>
</tr>
<tr>
<td><strong>Byte</strong></td>
<td><strong>Bit 7</strong></td>
</tr>
<tr>
<td>DBB3000</td>
<td>Positioning axes active</td>
</tr>
<tr>
<td>DBB3001</td>
<td>-</td>
</tr>
<tr>
<td>DBB3002</td>
<td>-</td>
</tr>
<tr>
<td>DBB3003</td>
<td>-</td>
</tr>
</tbody>
</table>
The following requirements must be satisfied in order to use the functions listed below:

- The axis or spindle is correctly assigned to the PLC.
- Controller and pulse enable are set.
- After setting all of the control signals, only one of the start signals is set in DB380x.DBB3000.

### 14.6.4.4 Spindle positioning

<table>
<thead>
<tr>
<th>DB380x</th>
<th>PLC → NCK control signals</th>
<th>Valid values</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3002.0</td>
<td>Incremental</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3002.1</td>
<td>Shortest path</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3003.0</td>
<td>Absolute, negative direction</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3003.1</td>
<td>Absolute, positive direction</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBD3004</td>
<td>Setpoint position/setpoint distance</td>
<td>REAL</td>
<td>If = 0, the value from MD35300 $MA_SPIND_POSCTRL_VELO (position control activation speed) is taken</td>
</tr>
<tr>
<td>DBD3008</td>
<td>Feedrate</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>DBX3000.6</td>
<td>Start</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX2.2</td>
<td>Delete distance-to-go, spindle reset</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB390x</th>
<th>NCK → PLC status signals</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3000.7</td>
<td>Positioning axes active</td>
<td>Also 1 when override = 0 or position setpoint reached when start = 1</td>
</tr>
<tr>
<td>DBX3000.6</td>
<td>Position reached</td>
<td>1: Position setpoint reached with &quot;Exact stop fine&quot;</td>
</tr>
<tr>
<td>DBX3000.0</td>
<td>Spindle cannot be started</td>
<td></td>
</tr>
<tr>
<td>DBX3000.1</td>
<td>Error while traversing</td>
<td>1: Error during traversing, evaluate error number in DBB3003!</td>
</tr>
<tr>
<td>DBB3003</td>
<td>Error number</td>
<td></td>
</tr>
<tr>
<td>DBX1.4</td>
<td>Axis/spindle stationary</td>
<td>1: if n &lt; n_{min}</td>
</tr>
</tbody>
</table>
14.6.4.5 Rotate spindle

<table>
<thead>
<tr>
<th>DB380x</th>
<th>PLC → NCK control signals</th>
<th>Valid values</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3002.0</td>
<td>Incremental</td>
<td>0</td>
<td>Direction of rotation as for M4:</td>
</tr>
<tr>
<td>DBX3002.1</td>
<td>Shortest distance</td>
<td>0</td>
<td>1: Direction of rotation specified by M4</td>
</tr>
<tr>
<td>DBX3002.5</td>
<td>Direction of rotation as for M4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3003.0</td>
<td>Absolute, negative direction</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBX3003.1</td>
<td>Absolute, positive direction</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBD3008</td>
<td>Feedrate velocity</td>
<td>REAL</td>
<td>Spindle speed</td>
</tr>
</tbody>
</table>
### PLC → NCK control signals

| DBX3000.5 | Start spindle rotation | 0|1 |
| DBX3001.5 | Stop spindle rotation  | 0|1 |

### NCK → PLC status signals

| DBX3000.7 | Positioning axes active | 1: for start or stop == 1 |
| DBX3000.6 | Position reached         | 1: Function was started without an error |
| DBX3000.1 | Error                    | 1: Error when traversing, evaluate error number in DBB3003! |
| DBB3003   | Error number             |
| DBX1.4    | Axis/spindle stationary  |

---

① Function activated by user with a positive edge of Start.
② Signals Positioning axis active and Position reached are signaled back, Position reached is in this case irrelevant.
③ The user stops spindle rotation by resetting Start and setting Stop.
④ The spindle stops and the Spindle stationary signal is set.
⑤ The user then resets Stop.
⑥ Reset of Stop causes Positioning axis active to be reset.
### Oscillate spindle

<table>
<thead>
<tr>
<th>DB380x</th>
<th>PLC → NCK control signals</th>
<th>Valid values</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3002.0</td>
<td>Incremental</td>
<td>0</td>
<td>It is not permissible that any of the bits are set.</td>
</tr>
<tr>
<td>DBX3002.1</td>
<td>Shortest distance</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBX3002.5</td>
<td>Direction of rotation as for M4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBX3003.0</td>
<td>Absolute, negative direction</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBX3003.1</td>
<td>Absolute, positive direction</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBD3004</td>
<td>Setpoint gear stage: Values 0, 1, 2, 3, 4, 5</td>
<td>REAL</td>
<td>MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 0 0-5: Oscillation MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 1 0: Oscillation 1: Oscillation with gear stage change M41 2: Oscillation with gear stage change M42 3: Oscillation with gear stage change M43 4: Oscillation with gear stage change M44 5: Oscillation with gear stage change M45</td>
</tr>
<tr>
<td>DBD3008</td>
<td>Feedrate velocity</td>
<td>REAL</td>
<td>When oscillating, no significance! The oscillation speed is taken from machine data MD35400, $MA_SPIND_OSCILL_DES_VELO.</td>
</tr>
<tr>
<td>DBX3000.5</td>
<td>Start spindle oscillation</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3001.5</td>
<td>Stop spindle oscillation</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB390x</th>
<th>NCK → PLC status signals</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3000.7</td>
<td>Positioning axes active</td>
<td>1: for start or stop == 1</td>
</tr>
<tr>
<td>DBX3000.6</td>
<td>Position reached</td>
<td>1: after start</td>
</tr>
<tr>
<td>DBX3000.1</td>
<td>Error</td>
<td>1: Error when traversing, evaluate error number in DBB3003!</td>
</tr>
<tr>
<td>DBX3000.0</td>
<td>Axis cannot be started</td>
<td>1: Error when starting, evaluate error number in DBB3003!</td>
</tr>
<tr>
<td>DBB3003</td>
<td>Error number</td>
<td></td>
</tr>
<tr>
<td>DBX1.4</td>
<td>Axis/spindle stationary</td>
<td></td>
</tr>
</tbody>
</table>
14.6.4.7 Indexing axis

<table>
<thead>
<tr>
<th>DB380x</th>
<th>PLC → NCK control signals</th>
<th>Valid values</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3002.0</td>
<td>Incremental</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3002.1</td>
<td>Shortest distance</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3002.5</td>
<td>Direction of rotation as for M4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBX3003.0</td>
<td>Absolute, negative direction</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3003.1</td>
<td>Absolute, positive direction</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBD3003.7</td>
<td>Indexing position</td>
<td>1</td>
<td>Indexing axis ON</td>
</tr>
<tr>
<td>DBD3004</td>
<td>Setpoint position/setpoint distance</td>
<td>DINT</td>
<td>for &quot;incremental&quot;: Setpoint distance</td>
</tr>
<tr>
<td>DB380x</td>
<td>PLC → NCK control signals</td>
<td>Valid values</td>
<td>Remark</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------</td>
<td>--------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DBD3008</td>
<td>Feedrate velocity</td>
<td>REAL</td>
<td>if = 0, the value is taken from machine data POS_AX_VE-LO (unit as set in machine data).</td>
</tr>
<tr>
<td>DBX3000.7</td>
<td>Start positioning axis</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX2.2</td>
<td>Delete distance-to-go, spindle reset</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB390x</th>
<th>NCK → PLC status signals</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3000.7</td>
<td>Positioning axes active</td>
<td>Also 1, if override = 0 or setpoint position reached.</td>
</tr>
<tr>
<td>DBX3000.6</td>
<td>Position reached</td>
<td>1: Setpoint position reached with &quot;Exact stop fine&quot;.</td>
</tr>
<tr>
<td>DBX3000.1</td>
<td>Error</td>
<td>1: Error when traversing, evaluate error number in DBB3003!</td>
</tr>
<tr>
<td>DBB3003</td>
<td>Error number</td>
<td></td>
</tr>
<tr>
<td>DBX1.4</td>
<td>Axis/spindle stationary</td>
<td></td>
</tr>
</tbody>
</table>

① Function activated by user with a positive edge of Start.  
**Note:** This is only possible when the Positioning axis active signal is reset!  
② Signals Positioning axis active and Position reached are signaled back, Position reached is in this case irrelevant.  
③ The user stops spindle oscillation by resetting Start and setting Stop.  
④ The spindle stops and the Spindle stationary signal is set.  
⑤ The user then resets Stop.  
⑥ Reset of Stop causes Positioning axis active to be reset.
### 14.6.4.8 Positioning axis metric

<table>
<thead>
<tr>
<th>DB380x</th>
<th>PLC → NCK control signals</th>
<th>Valid values</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3002.0</td>
<td>Incremental</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3002.1</td>
<td>Shortest distance</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3002.5</td>
<td>Direction of rotation as for M4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBX3003.0</td>
<td>Absolute, negative direction</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3003.1</td>
<td>Absolute, positive direction</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBD3002.2</td>
<td>Traversing dimension inch</td>
<td>0</td>
<td>Traversing dimension, metric</td>
</tr>
<tr>
<td>DBD3002.3</td>
<td>Handwheel override</td>
<td>0</td>
<td>Override OFF</td>
</tr>
<tr>
<td>DBD3004</td>
<td>Setpoint position/setpoint distance</td>
<td>REAL</td>
<td>for &quot;incremental&quot;: Setpoint distance</td>
</tr>
<tr>
<td>DBD3008</td>
<td>Feedrate velocity</td>
<td>REAL</td>
<td>if = 0, the value is taken from machine data POS_AX_VE-LO (unit as set in machine data).</td>
</tr>
<tr>
<td>DBX3000.7</td>
<td>Start positioning axis</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX2.2</td>
<td>Delete distance-to-go, spindle reset</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB390x</th>
<th>NCK → PLC status signals</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3000.7</td>
<td>Positioning axes active</td>
<td>Also 1, if override = 0 or setpoint position reached.</td>
</tr>
<tr>
<td>DBX3000.6</td>
<td>Position reached</td>
<td>1: Axis has reached the setpoint position.</td>
</tr>
<tr>
<td>DBX3000.1</td>
<td>Error</td>
<td>1: Error when traversing, evaluate error number in DBB3003!</td>
</tr>
<tr>
<td>DBBB3003</td>
<td>Error number</td>
<td></td>
</tr>
</tbody>
</table>
① First function activation using positive edge of Start.
② Positioning axis active = 1 shows that the function is active and that the output signals are valid, Position reached and Axis stationary are possibly withdrawn.
③ Positive acknowledgement Position reached = 1 and Positioning axis active = 1
④ Reset of function activation after receipt of acknowledgment
⑤ Signal change via function
⑥ Positioning is interrupted by delete distance to go, signal duration min. 1 PLC cycle.
⑦ The signals Position reached and Error are reset, the Error number can be read (in this case, 30).

14.6.4.9 Positioning axis inch

<table>
<thead>
<tr>
<th>DB380x</th>
<th>PLC → NCK control signals</th>
<th>Valid values</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3002.0</td>
<td>Incremental</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3002.1</td>
<td>Shortest distance</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3002.5</td>
<td>Direction of rotation as for M4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBX3003.0</td>
<td>Absolute, negative direction</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3003.1</td>
<td>Absolute, positive direction</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBD3002.2</td>
<td>Traversing dimension inch</td>
<td>1</td>
<td>Traversing dimension inch</td>
</tr>
<tr>
<td>DBD3002.3</td>
<td>Handwheel override</td>
<td>0</td>
<td>Override OFF</td>
</tr>
<tr>
<td>DBD3004</td>
<td>Setpoint position/setpoint distance</td>
<td>REAL</td>
<td>for &quot;incremental&quot;: Setpoint distance</td>
</tr>
<tr>
<td>DBD3008</td>
<td>Feedrate velocity</td>
<td>REAL</td>
<td>if = 0, the value is taken from machine data POS_AX_VELO (unit as set in machine data).</td>
</tr>
</tbody>
</table>
### PLC → NCK control signals

<table>
<thead>
<tr>
<th>DBX3000.7</th>
<th>Start positioning axis</th>
<th>Valid values</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX2.2</td>
<td>Delete distance-to-go, spindle reset</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### NCK → PLC status signals

<table>
<thead>
<tr>
<th>DBX3000.7</th>
<th>Positioning axes active</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3000.6</td>
<td>Position reached</td>
<td>1: Axis has reached the setpoint position.</td>
</tr>
<tr>
<td>DBX3000.1</td>
<td>Error</td>
<td>1: Error when traversing, evaluate error number in DBB3003!</td>
</tr>
<tr>
<td>DBB3003</td>
<td>Error number</td>
<td></td>
</tr>
</tbody>
</table>

---

**Diagram Explanation:***

1. First function activation using positive edge of *Start*.
2. *Positioning axis active* = 1 shows that the function is active and that the output signals are valid, *Position reached* and *Axis stationary* are possibly withdrawn.
3. Positive acknowledgement *Position reached* = 1 and *Positioning axis active* = 1
4. Reset of function activation after receipt of acknowledgment
5. Signal change via function
6. Positioning is interrupted by delete distance to go, signal duration min. 1 PLC cycle.
7. The signals *Position reached* and *Error* are reset, the *Error number* can be read (in this case, 30).
### 14.6.4.10 Positioning axis metric with handwheel override

<table>
<thead>
<tr>
<th>DB380x</th>
<th>PLC → NCK control signals</th>
<th>Valid values</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3002.0</td>
<td>Incremental</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3002.1</td>
<td>Shortest distance</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3002.5</td>
<td>Direction of rotation as for M4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBX3003.0</td>
<td>Absolute, negative direction</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3003.1</td>
<td>Absolute, positive direction</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBD3002.2</td>
<td>Traversing dimension inch</td>
<td>0</td>
<td>Traversing dimension, metric</td>
</tr>
<tr>
<td>DBD3002.3</td>
<td>Handwheel override</td>
<td>1</td>
<td>Override ON</td>
</tr>
<tr>
<td>DBD3004</td>
<td>Setpoint position/setpoint distance</td>
<td>REAL</td>
<td>for &quot;incremental&quot;: Setpoint distance</td>
</tr>
<tr>
<td>DBD3008</td>
<td>Feedrate velocity</td>
<td>REAL</td>
<td>if = 0, the value is taken from machine data POS_AX_VELO (unit as set in machine data).</td>
</tr>
<tr>
<td>DBX3000.7</td>
<td>Start positioning axis</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX2.2</td>
<td>Delete distance-to-go, spindle reset</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB390x</th>
<th>NCK → PLC status signals</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3000.7</td>
<td>Positioning axes active</td>
<td>Also 1, if override = 0 or setpoint position reached.</td>
</tr>
<tr>
<td>DBX3000.6</td>
<td>Position reached</td>
<td>1: Axis has reached the setpoint position.</td>
</tr>
<tr>
<td>DBX3000.1</td>
<td>Error</td>
<td>1: Error when traversing, evaluate error number in DBB3003!</td>
</tr>
<tr>
<td>DBB3003</td>
<td>Error number</td>
<td></td>
</tr>
</tbody>
</table>
### 14.6.4.11 Positioning axis inch with handwheel override

<table>
<thead>
<tr>
<th>DB380x</th>
<th>PLC → NCK control signals</th>
<th>Valid values</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3002.0</td>
<td>Incremental</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3002.1</td>
<td>Shortest distance</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3002.5</td>
<td>Direction of rotation as for M4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBX3003.0</td>
<td>Absolute, negative direction</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3003.1</td>
<td>Absolute, positive direction</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBD3002.2</td>
<td>Traversing dimension inch</td>
<td>1</td>
<td>Traversing dimension inch</td>
</tr>
<tr>
<td>DBD3002.3</td>
<td>Handwheel override</td>
<td>1</td>
<td>Override ON</td>
</tr>
<tr>
<td>DBD3004</td>
<td>Setpoint position/setpoint distance</td>
<td>REAL</td>
<td>for &quot;incremental&quot;: Setpoint distance</td>
</tr>
<tr>
<td>DBD3008</td>
<td>Feedrate velocity</td>
<td>REAL</td>
<td>if = 0, the value is taken from machine data POS_AX_VELO (unit as set in machine data).</td>
</tr>
<tr>
<td>DB380x</td>
<td>PLC → NCK control signals</td>
<td>Valid values</td>
<td>Remark</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------</td>
<td>--------------</td>
<td>--------</td>
</tr>
<tr>
<td>DBX3000.7</td>
<td>Start positioning axis</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX2.2</td>
<td>Delete distance-to-go, spindle reset</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB390x</th>
<th>NCK → PLC status signals</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3000.7</td>
<td>Positioning axes active</td>
<td>Also 1, if override = 0 or setpoint position reached.</td>
</tr>
<tr>
<td>DBX3000.6</td>
<td>Position reached</td>
<td>1: Axis has reached the setpoint position.</td>
</tr>
<tr>
<td>DBX3000.1</td>
<td>Error</td>
<td>1: Error when traversing, evaluate error number in DBB3003!</td>
</tr>
<tr>
<td>DBB3003</td>
<td>Error number</td>
<td></td>
</tr>
</tbody>
</table>

---

**Diagram:***

<table>
<thead>
<tr>
<th>to PLC axis:</th>
<th>from PLC axis:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Positioning axes active</td>
</tr>
<tr>
<td>Delete distance-to-go</td>
<td>Position reached</td>
</tr>
<tr>
<td></td>
<td>Error</td>
</tr>
</tbody>
</table>

1. First function activation using positive edge of *Start.*
2. *Positioning axis active* = 1 shows that the function is active and that the output signals are valid, *Position reached* and *Axis stationary* are possibly withdrawn.
3. Positive acknowledgement *Position reached* = 1 and *Positioning axis active* = 1
4. Reset of function activation after receipt of acknowledgment
5. Signal change via function
6. Positioning is interrupted by delete distance to go, signal duration min. 1 PLC cycle.
7. The signals *Position reached* and *Error* are reset, the *Error number* can be read (in this case, 30).
## 14.6.4.12 Rotate spindle with automatic gear stage selection

<table>
<thead>
<tr>
<th>DB380x</th>
<th>PLC → NCK control signals</th>
<th>Valid values</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3002.0</td>
<td>Incremental</td>
<td>0</td>
<td>Direction of rotation as for M4:</td>
</tr>
<tr>
<td>DBX3002.1</td>
<td>Shortest distance</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBX3002.5</td>
<td>Direction of rotation as for M4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3003.0</td>
<td>Absolute, negative direction</td>
<td>0</td>
<td>0: Direction of rotation specified by M3</td>
</tr>
<tr>
<td>DBX3003.1</td>
<td>Absolute, positive direction</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBD3002.7</td>
<td>Automatic gear stage selection</td>
<td>1</td>
<td>Automatic gear stage selection ON</td>
</tr>
<tr>
<td>DBD3008</td>
<td>Feedrate velocity</td>
<td>REAL</td>
<td>Spindle speed</td>
</tr>
<tr>
<td>DBD3000.5</td>
<td>Start spindle rotation</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3001.5</td>
<td>Stop spindle rotation</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB390x</th>
<th>NCK → PLC status signals</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3000.7</td>
<td>Positioning axes active</td>
<td>1: for start or stop == 1,</td>
</tr>
<tr>
<td>DBX3000.6</td>
<td>Position reached</td>
<td>1: Setpoint speed is output</td>
</tr>
<tr>
<td>DBX3000.1</td>
<td>Error</td>
<td>1: Error when traversing, evaluate error number in DBB3003!</td>
</tr>
<tr>
<td>DBB3003</td>
<td>Error number</td>
<td></td>
</tr>
<tr>
<td>DBX1.4</td>
<td>Axis/spindle stationary</td>
<td></td>
</tr>
</tbody>
</table>
14.6.4.13 Rotate spindle with constant cutting rate [m/min]

<table>
<thead>
<tr>
<th>DB380x</th>
<th>PLC → NCK control signals</th>
<th>Valid values</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX3002.0</td>
<td>Incremental</td>
<td>0</td>
<td>Direction of rotation as for M4:</td>
</tr>
<tr>
<td>DBX3002.1</td>
<td>Shortest distance</td>
<td>0</td>
<td>1: Direction of rotation specified by M4</td>
</tr>
<tr>
<td>DBX3002.5</td>
<td>Direction of rotation as for M4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3003.0</td>
<td>Absolute, negative direction</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBX3003.1</td>
<td>Absolute, positive direction</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DBD3002.2</td>
<td>Traversing dimension inch</td>
<td>0</td>
<td>Traversing dimension, metric</td>
</tr>
<tr>
<td>DBD3002.6</td>
<td>Const. Cutting rate</td>
<td>1</td>
<td>Constant cutting rate ON</td>
</tr>
<tr>
<td>DBD3008</td>
<td>Feedrate velocity</td>
<td>REAL</td>
<td>Spindle speed</td>
</tr>
<tr>
<td>DBD3000.5</td>
<td>Start spindle rotation</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DBX3001.5</td>
<td>Stop spindle rotation</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DB390x</td>
<td>NCK → PLC status signals</td>
<td>Remark</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>DBX3000.7</td>
<td>Positioning axes active</td>
<td>1: for start or stop == 1,</td>
<td></td>
</tr>
<tr>
<td>DBX3000.6</td>
<td>Position reached</td>
<td>1: Setpoint speed is output</td>
<td></td>
</tr>
<tr>
<td>DBX3000.1</td>
<td>Error</td>
<td>1: Error when traversing, evaluate error number in DBB3003!</td>
<td></td>
</tr>
<tr>
<td>DBB3003</td>
<td>Error number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBX1.4</td>
<td>Axis/spindle stationary</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

① Function activated by user with a positive edge of Start.
② Signals Positioning axis active and Position reached are signaled back, Position reached is in this case irrelevant.
③ The user stops spindle rotation by resetting Start and setting Stop.
④ The spindle stops and the Spindle stationary signal is set.
⑤ The user then resets Stop.
⑥ Reset of Stop causes Positioning axis active to be reset.
14.6.4.14 Rotate spindle with constant cutting rate [feet/min]

<table>
<thead>
<tr>
<th>DBX3002.0</th>
<th>PLC → NCK control signals</th>
<th>Valid values</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental</td>
<td>PLC to NCK control signals</td>
<td>0</td>
<td>Direction of rotation as for M4:</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBX3002.1</td>
<td>Shortest distance</td>
<td>0</td>
<td>1: Direction of rotation specified by M4</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBX3002.5</td>
<td>Direction of rotation as for M4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBX3003.0</td>
<td>Absolute, negative direction</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBX3003.1</td>
<td>Absolute, positive direction</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBD3002.2</td>
<td>Traversing dimension inch</td>
<td>1</td>
<td>Traversing dimension inch</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBD3002.6</td>
<td>Const. Cutting rate</td>
<td>1</td>
<td>Constant cutting rate ON</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBD3008</td>
<td>Feedrate velocity</td>
<td>REAL</td>
<td>Spindle speed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBD3000.5</td>
<td>Start spindle rotation</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBX3001.5</td>
<td>Stop spindle rotation</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DBX3000.7</th>
<th>NCK → PLC status signals</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positioning axes active</td>
<td>PLC status signals</td>
<td>1: for start or stop == 1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBX3000.6</td>
<td>Position reached</td>
<td>1: Setpoint speed is output</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBX3000.1</td>
<td>Error</td>
<td>1: Error when traversing, evaluate error number in DBB3003!</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBB3003</td>
<td>Error number</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBX1.4</td>
<td>Axis/spindle stationary</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
14.6.4.15 Error messages

If a function could not be executed, the following signals are set depending on the error:

- DB390x .DBX3000.0 == 1 (axis cannot be started)
- DB390x .DBX3000.1 == 1 (error during travel)

The exact error cause is indicated as:
- DB390x .DBB3003 (error number)

<table>
<thead>
<tr>
<th>Error number</th>
<th>Decimal</th>
<th>Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01</td>
<td>01</td>
<td>Several functions of the axis/spindle were activated simultaneously</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>14</td>
<td>A function was started without the position being reached</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>1E</td>
<td>The axis/spindle was transferred to the NC while still in motion</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>28</td>
<td>The axis is programmed by the NC program, NCK internal error</td>
</tr>
</tbody>
</table>

1 Function activated by user with a positive edge of Start.

2 Signals Positioning axis active and Position reached are signaled back, Position reached is in this case irrelevant.

3 The user stops spindle rotation by resetting Start and setting Stop.

4 The spindle stops and the Spindle stationary signal is set.

5 The user then resets Stop.

6 Reset of Stop causes Positioning axis active to be reset.
<table>
<thead>
<tr>
<th>Error number</th>
<th>Decimal</th>
<th>Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>32</td>
<td></td>
<td>Permanently assigned PLC axis: Traverses (JOG) or is referencing</td>
</tr>
<tr>
<td>60</td>
<td>3C</td>
<td></td>
<td>Permanently assigned PLC axis: Channel status does not permit a start</td>
</tr>
<tr>
<td>100</td>
<td>64</td>
<td></td>
<td>False position programmed for axis/spindle (alarm number 1) 16830</td>
</tr>
<tr>
<td>101</td>
<td>65</td>
<td></td>
<td>Programmed speed is too high</td>
</tr>
<tr>
<td>102</td>
<td>66</td>
<td></td>
<td>Incorrect value range for constant cutting rate (alarm number 1) 14840</td>
</tr>
<tr>
<td>104</td>
<td>68</td>
<td></td>
<td>Following spindle: Illegal programming (alarm number 1) 22030</td>
</tr>
<tr>
<td>105</td>
<td>69</td>
<td></td>
<td>No measuring system available (alarm number 1) 16770</td>
</tr>
<tr>
<td>106</td>
<td>6A</td>
<td></td>
<td>Positioning process of the axis still active (alarm number 1) 22052</td>
</tr>
<tr>
<td>107</td>
<td>6B</td>
<td></td>
<td>Reference mark not found (alarm number 1) 22051</td>
</tr>
<tr>
<td>108</td>
<td>6C</td>
<td></td>
<td>No transition from speed control to position control (alarm number 1) 22050</td>
</tr>
<tr>
<td>109</td>
<td>6D</td>
<td></td>
<td>Reference mark not found (alarm number 1) 22051</td>
</tr>
<tr>
<td>110</td>
<td>6E</td>
<td></td>
<td>Velocity/speed is negative</td>
</tr>
<tr>
<td>111</td>
<td>6F</td>
<td></td>
<td>Setpoint speed is zero</td>
</tr>
<tr>
<td>112</td>
<td>70</td>
<td></td>
<td>Invalid gear stage</td>
</tr>
<tr>
<td>115</td>
<td>73</td>
<td></td>
<td>Programmed position has not been reached</td>
</tr>
<tr>
<td>117</td>
<td>75</td>
<td></td>
<td>G96/G961 is not active in the NC</td>
</tr>
<tr>
<td>118</td>
<td>76</td>
<td></td>
<td>G96/G961 is still active in the NC</td>
</tr>
<tr>
<td>120</td>
<td>78</td>
<td></td>
<td>Axis is not an indexing axis (alarm number 1) 20072</td>
</tr>
<tr>
<td>121</td>
<td>79</td>
<td></td>
<td>Indexing position error (alarm number 1) 17510</td>
</tr>
<tr>
<td>125</td>
<td>7D</td>
<td></td>
<td>DC (shortest distance) not possible (alarm number 1) 16800</td>
</tr>
<tr>
<td>126</td>
<td>7E</td>
<td></td>
<td>Absolute value minus not possible (alarm number 1) 16820</td>
</tr>
<tr>
<td>127</td>
<td>7F</td>
<td></td>
<td>Absolute value plus not possible (alarm number 1) 16810</td>
</tr>
<tr>
<td>128</td>
<td>80</td>
<td></td>
<td>No transverse axis available for diameter programming (alarm number 1) 16510</td>
</tr>
<tr>
<td>130</td>
<td>82</td>
<td></td>
<td>Software limit switch plus (alarm number 1) 20070</td>
</tr>
<tr>
<td>131</td>
<td>83</td>
<td></td>
<td>Software limit switch minus (alarm number 1) 20070</td>
</tr>
<tr>
<td>132</td>
<td>84</td>
<td></td>
<td>working area limitation plus (alarm number 1) 20071</td>
</tr>
<tr>
<td>133</td>
<td>85</td>
<td></td>
<td>working area limitation minus (alarm number 1) 20071</td>
</tr>
<tr>
<td>134</td>
<td>85</td>
<td></td>
<td>Frame not permitted for indexing axis</td>
</tr>
<tr>
<td>135</td>
<td>87</td>
<td></td>
<td>Indexing axis with &quot;Hirth toothing&quot; is active (alarm number 1) 17501</td>
</tr>
<tr>
<td>136</td>
<td>88</td>
<td></td>
<td>Indexing axis with &quot;Hirth toothing&quot; is active and axis not referenced (alarm number 1) 17503</td>
</tr>
<tr>
<td>137</td>
<td>89</td>
<td></td>
<td>Spindle operation not possible for transformed spindle/axis (alarm number 1) 22290</td>
</tr>
<tr>
<td>138</td>
<td>8A</td>
<td></td>
<td>The corresponding effective coordinate-system-specific working area limit plus violated for the axis (alarm number 1) 20082</td>
</tr>
<tr>
<td>139</td>
<td>8B</td>
<td></td>
<td>The corresponding effective coordinate-system-specific working area limit minus violated for the axis (alarm number 1) 20082</td>
</tr>
<tr>
<td>200</td>
<td>C8</td>
<td></td>
<td>System alarm number 1) 450007</td>
</tr>
</tbody>
</table>

1) The detailed alarm description is contained in: Alarms diagnostics manual; SINUMERIK 828D, SINAMICS S120
14.6.5 Starting ASUBs

14.6.5.1 General

From the PLC, interrupt 1 or 2 can be initiated on the NC, in principle, at any point in time. As a consequence, the program currently being processed in the NC is interrupted, and the interrupt program assigned to the interrupt executed:

- Interrupt 1: _N_CMD_DIR/PLCASUP1.SPF
- Interrupt 2: _N_CMD_DIR/PLCASUP2.SPF

Note

A start signal must not be set in the following cases:

- PI service "ASUB" has not been completed yet.
- DB2600.DBX0.1 == 1 (emergency stop)
- The request "Channel reset" is still active in the channel. The ASUB can only be started again when the channel is in the "Reset" state AND DB3300.DBX3.7 == 1 (channel state reset).

See also

PI service: ASUB (Page 942)

14.6.5.2 Job start

<table>
<thead>
<tr>
<th>DB3400</th>
<th>ASUB: Job [r/w]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PLC → NCK interface</td>
</tr>
<tr>
<td>Byte</td>
<td>Bit 7</td>
</tr>
<tr>
<td>DBB 0000</td>
<td>-</td>
</tr>
<tr>
<td>DBB 0001</td>
<td>-</td>
</tr>
<tr>
<td>DBB 0002</td>
<td>-</td>
</tr>
<tr>
<td>DBB 0003</td>
<td>-</td>
</tr>
</tbody>
</table>

Start INT1:
- Bit 0 = 1: Request to start the ASUB, which is assigned to INT1.
- Bit 0 = 0: Reset the ASUB request after acknowledgement in the result interface DBB1000.

Start INT2
- Bit 0 = 1: Request to start the ASUB, which is assigned to INT2.
- Bit 0 = 0: Reset the ASUB request after acknowledgement in the result interface DBB1001.
14.6.5.3 Job result

<table>
<thead>
<tr>
<th>DB3400</th>
<th>ASUB: Result [r]</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCK → PLC interface</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte</th>
<th>INT1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBB 1000</td>
<td>SUB can-not be executed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBB 1001</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

| DBB 1002 |
| DBB 2003 |

1) Negative acknowledgement: e.g. for emergency stop or channel reset request.
2) Negative acknowledgement; Number has not been assigned yet. Remedy: Execute PI service "ASUB".
3) Positive acknowledgement; ASUB successfully completed. Reset start signal:

- DB3400.DBX1000.0 == 1 ⇒ DB3400.DBX0000.0 = 0
- DB3400.DBX1001.0 == 1 ⇒ DB3400.DBX0001.0 = 0
14.6.5.4 Signal flow

Signal flow

Figure 14-2 Example: Signal flow

1. Function activated by user with a positive edge of Start.
2. ASUB is being executed is signaled back.
3. The acknowledgement ASUB completed indicates the successful execution, ASUB is being executed is withdrawn.
4. The signal to initiate the function is reset after receiving the acknowledgement from the user.
5. Signal change by the firmware.
6. Not permitted! If function activation is reset prior to receipt of acknowledgement, the output signals are not updated – without the operational sequence of the activated function being affected.
7. ASUB cannot be executed: Negative acknowledgement, error occurred.

Figure 14-2 Example: Signal flow
14.6 Function interface
15.1 Brief Description

Function

When referencing a machine axis, the coordinate system of the machine axis is synchronized with the coordinate system of the machine. The machine axis is traversed to the machine zero and then the actual position of the machine axis is set to zero.

If the machine zero cannot be directly approached as a result of the machine design, then a reference point is defined in the traversing range of the machine axis, which is then used to synchronize the machine axis. Its position with reference to the machine zero must be known. When referencing, the actual machine axis position is set to this value.

Measuring systems and referencing methods

When referencing, machine axes can be synchronized with the following measuring systems and referencing types:

- Measuring systems
  - Incremental rotary measuring system with at least one zero mark
  - Incremental linear measuring system
  - Rotary measuring system with distance-coded reference marks (supplied by Heidenhain)
  - Linear measuring system with distance-coded reference marks (supplied by Heidenhain)
  - Absolute rotary measuring system
  - Absolute linear measuring system

- Referencing methods
  - Referencing with incremental measuring systems with BERO and one-edge and two-edge detection
  - Referencing with incremental measuring systems with replacement of homing cam with BERO
  - Referencing with incremental measuring systems with BERO with configured approach velocity for spindle applications
  - Referencing with measuring systems with distance-coded reference marks by overtravelling 2 or 4 zero marks
  - Referencing of passive measuring systems using measuring system adjustment
  - Referencing in follow-up mode
  - Referencing with cam switch at the drive
Start

Referencing the machine axis can be manually started, or from the part program:

- Manual: Operation mode JOG and MDI, machine function REF
- Part program: Command G74

15.2 Axis-specific referencing

For axis-specific referencing using the reference point approach, the operation must be individually started for each machine axis to be referenced.

Selecting mode and machine function

Before starting reference point approach of the machine axes, you must first place the relevant mode group in JOG or MDI mode:

DB11, ... DBX0.2 (active JOG mode)
DB11, ... DBX0.1 (active MDI mode)

Then machine function REF (reference point approach) must be selected:

DB11, ... DBX1.2 (REF machine function)

Start of reference point approach

In axis-specific reference point approach, each machine axis must be started individually.

Reference point approach is started with the axis-specific traversing keys:

DB31, ... DBX4.6 (Traversing key minus)
DB31, ... DBX4.7 (Traversing key minus)

Direction enable

To avoid operator errors, the direction release must be parameterized. The direction enable specifies which traversing key starts the reference point approach:

MD34010 $MA_REFP_CAM_DIR_IS_MINUS = <value>

<table>
<thead>
<tr>
<th>&lt;value&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reference point approach in plus direction</td>
</tr>
<tr>
<td>1</td>
<td>Reference point approach in minus direction</td>
</tr>
</tbody>
</table>

Jog mode

The following machine data element can be used to specify whether reference point approach is completed when the direction key is pressed once or whether the operator is required to keep the direction key pressed (jogging) for safety reasons:

MD11300 $MN_JOG_INC_MODE_LEVELTRIGGRD (INC and REF in jog mode)
If the machine operator releases the direction key, the machine axis is decelerated to zero speed. Reference point approach is not aborted. Reference point approach is continued the next time the direction key is pressed.

Referencing status

The referencing status of the machine axis is reset with the start of the reference point approach:

- DB31, ..., DBX60.4 (referenced / synchronized 1)
- DB31, ..., DBX60.5 (referenced / synchronized 2)
- DB21, ..., DBX36.2 (all axes with obligatory reference point are referenced)

Distance-coded measuring systems

In distance-coded measuring systems, reference point approach can be started with any traversing key.

Sequence

The machine operator or machine manufacturer (via the PLC user program) is responsible for ensuring that the machine axes are referenced in the proper order.

- Machine operator
  The machine axes must be started by the machine operator in the specified order.

- Machine manufacturer
  The PLC user program of the machine manufacturer allows machine axes to be started only in the proper order.

Simultaneous reference point approach of several machine axes

Several machine axes can be referenced simultaneously, depending on the control:

Completion of reference point approach

Acknowledgment that reference point approach of a machine axis has been successfully completed is given by setting the referencing status:

- DB31, ..., DBX60.4 (referenced / synchronized 1)
- DB31, ..., DBX60.5 (referenced / synchronized 2)

Cancellation of reference point approach

In axis-specific reference point approach, the machine axis is traversed in the channel that was assigned as the master channel of the machine axis.

MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN

For aborting the reference point approach, either mode group reset or channel reset for the master channel of the machine axis must be activated:
DB11, ... DBX0.7 (mode group reset)
DB21, ... DBX7.7 (channel reset)

All machine axes that have not yet successfully completed reference point approach when the action is cancelled remain in status "Not referenced":
DB31, ... DBX60.4 (referenced / synchronized 1)
DB31, ... DBX60.5 (referenced / synchronized 2)

15.3 Channelspecific referencing

For channel-specific referencing, all machine axes of the channel are referenced in the parameterized sequence when reference point approach is initiated.

Selecting mode and machine function

Before starting the reference point approach of the machine axes, the associated mode group must be switched into the JOG or MDI operating mode:
DB11, ... DBX0.2 (active JOG mode)
DB11, ... DBX0.1 (active MDI mode)

Then machine function REF (reference point approach) must be selected:
DB11, ... DBX1.2 (REF machine function)

Parameterizing the axis sequence

The following machine data element is used to specify the sequence in which the machine axes of the channel are referenced:
MD34110 $MA_REFP_CYCLE_NR = <number>

<table>
<thead>
<tr>
<th>&lt;number&gt;</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>The machine axis does not have to be referenced for NC START in the channel.</td>
</tr>
<tr>
<td>0</td>
<td>The machine axis does not participate in channel-specific reference point approach.</td>
</tr>
<tr>
<td>1 - 15</td>
<td>Sequence number in channel-specific reference point approach.</td>
</tr>
</tbody>
</table>

The machine axes are referenced in ascending order of numbers.
Machine axes with the same number will be referenced simultaneously.

Simultaneous reference point approach of several machine axes

Several machine axes can be referenced simultaneously, depending on the control:

Start of reference point approach

Channel-specific reference point approach is started with:
DB21, ... DBX1.0 (activate referencing)
The status of channel-specific reference point approach is indicated by the channel with:
DB21, ... DBX33.0 (activate referencing)

Referencing status
The referencing status of the machine axis is reset with the start of the reference point approach:
DB31, ... DBX60.4 (referenced / synchronized 1)
DB31, ... DBX60.5 (referenced / synchronized 2)

Completion of reference point approach
As soon as channel-specific reference approach has been successfully completed for all machine axes involved, this is acknowledged with:
DB21, ... DBX36.2 (all axes with obligatory reference point are referenced)

Cancellation of reference point approach
In channel-specific reference point approach the machine axis is traversed in the channel to which that axis is currently assigned as channel axis.
For aborting the reference point approach either mode group reset or channel reset for the corresponding channel must be activated:
DB11, ... DBX0.7 (mode group reset)
DB21, ... DBX7.7 (channel reset)
All machine axes for which the reference point approach is not yet successfully completed when the action is cancelled remain in status "Not referenced":
DB31, ... DBX60.4 (referenced / synchronized 1)
DB31, ... DBX60.5 (referenced / synchronized 2)

15.4 Reference point approach from part program (G74)

Function
With command G74, machine axes can be referenced from a part program or synchronized action either for the first time or again.
Referencing must be repeated, for example, after:
- Actual value offset PRESETON
  References: Programming Guide Advanced, section "Coordinate transformations (Frames)" > "Preset offset with PRESETON"
- Parking a machine axis (Page 113)
- Exceeding the encoder limit frequency of the position measuring system
15.5 Referencing with incremental measurement systems

Programming

Syntax

G74 <machine axis> {<machine axis>}

Meaning

<table>
<thead>
<tr>
<th>G74:</th>
<th>Command for referencing machine axes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alone in the block:</td>
<td>Yes</td>
</tr>
<tr>
<td>Effective-ness:</td>
<td>Non-modal</td>
</tr>
</tbody>
</table>

| <machine axis>: | The name of a machine axis must be specified (MD10000 $MN_AXCONF_MACHAX_NAME_TAB). The machine axis must be a channel axis of the channel in which the part program or the synchronized action is executed. |

Reset response

Mode group reset or channel reset aborts the reference point approach for all programmed machine axes:

- DB11, ... DBX0.7 (mode group reset)
- DB21, ... DBX7.7 (channel reset)

All machine axes for which the reference point approach is not yet successfully completed when the action is canceled remain in status "Not referenced":

- DB31, ... DBX60.4 (referenced/synchronized 1) == 0
- DB31, ... DBX60.5 (referenced/synchronized 2) == 0

15.5 Referencing with incremental measurement systems

15.5.1 Hardware signals

Depending on the machine design and the properties of the incremental measuring system used, different hardware signals must be connected.

Reference cam

- Connection
  The reference cam signal can be connected to a digital input of an external PLC I/O module or to a fast input on the NCU X142 interface.

- NC/PLC interface signal
  The reference cam signal must be transferred from the PLC user program to the axial NC/PLC interface:
  DB31, ... DBX12.7 (deceleration of reference point approach)
Zero mark selection
If during the reference point approach of the axis or spindle several zero marks of the measuring system are detected (e.g. measuring gear between the motor and encoder), then the specific zero mark must be selected with an additional BERO signal.

- Connection
  The BERO must be connected to a fast digital input on the NCU X122 or X132 interface.

- Activation
  In order that the BERO signal is evaluated, the digital input to which the BERO is connected must be selected in drive parameter p0493 for the axis/spindle.

Equivalent zero mark
If the used measuring system does not provide a zero mark signal, an equivalent zero mark can be created via a BERO signal.

- Connection
  The BERO must be connected to a fast digital input on the NCU X122 or X132 interface.

- Activation
  In order that the BERO signal is evaluated, the digital input to which the BERO is connected must be selected in drive parameter p0494 or p0495 for the axis/spindle.

Overview

<table>
<thead>
<tr>
<th>Signal</th>
<th>Connection: Dig. input via</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference cam</td>
<td>Ext. PLC I/O module or NCU: X142</td>
<td>PLC user program: DB31, ... DBX12.7</td>
</tr>
<tr>
<td>Zero mark selection</td>
<td>NCU: X122 or X132</td>
<td>Drive parameter: p0493</td>
</tr>
<tr>
<td>External zero mark or equivalent zero mark</td>
<td>NCU: X122 or X132</td>
<td>Drive parameter: p0494 or p0495</td>
</tr>
</tbody>
</table>

References

- NCU interfaces: SINUMERIK 840D sl Manual, NCU7x0.3 PN, Section "Connecting" > "Digital I/Os"
- Drive parameters: SINAMICS S120/S150 List Manual
15.5.2 Zero mark selection with BERO

Function

Referencing of incremental measuring systems is based on the unique position of the encoder zero mark relative to the overall traversing range of the machine axis. If because of machine-specific conditions, several encoder zero marks are detected in the traversing range of the machine axis (for examples, see figure below), a BERO must be mounted on the machine for clear determination of the reference point. The position of the reference point is then derived from the combination of BERO signal and encoder zero mark.

Parameterization

**NC: Referencing mode**

"Referencing of incremental, rotary or linear measuring systems: Zero pulse on the encoder track" should be parameterized as referencing mode:

MD34200 $MA_ENC_REFP_MODE[<axis>] = 1

**Drive: Zero mark selection**

The digital input on the NCU interface to which the BERO is connected must be set in parameter p0493.

**Note**

BERO signal: Zero mark selection

The processing of the BERO signal is performed exclusively in the drive. Connection and parameterization, see Section "Hardware signals (Page 1154)".
15.5.3 **Time sequence**

Reference point approach with incremental measuring systems can be divided into three phases:

- Phase 1: "Phase 1: Traversing to the reference cam (Page 1158)"
- Phase 2: "Phase 2: Synchronization with the zero mark (Page 1160)"
- Phase 3: "Phase 3: Traversing to the reference point (Page 1165)"

![Diagram of time sequence](image)

Figure 15-2  Time sequence when referencing with incremental measuring systems (example)
15.5.4 Phase 1: Traversing to the reference cam

Phase 1: Graphic representation

Figure 15-3 Phase 1: Traversing to the reference cam

Phase 1: Start

To start the reference point approach, see Sections "Axisspecific referencing (Page 1150)" and "Axisspecific referencing (Page 1150)".

Phase 1: Sequence

In Phase 1, depending on the position of the machine axis with reference to the reference cam, we distinguish between three cases:

1. The machine axis is positioned before the reference cam
2. The machine axis is positioned on the reference cam
3. The machine axis has no reference cam
Case 1: The machine axis is positioned before the reference cam
After the start of reference point approach, the machine axis is accelerated in the parameterized direction and to the parameterized reference point approach velocity:
- MD34010 $MA_REFP_CAM_DIR_IS_MINUS (approach reference point in minus direction)
- MD34020 $MA_REFP_VELO_SEARCH_CAM (reference point approach velocity)
The reaching of the reference cam must detected by querying a digital input in the PLC user program and communicated to the NC via the following interface signal:
DB31, ... DBX12.7 = 1 (reference point approach deceleration)
With detection of the NC/PLC interface signal, the machine axis is decelerated to zero speed. Whereby at least the distance $s_{min}$ is traversed. This ensures that the machine axis leaves the reference cam in Phase 2 with the parameterized reference point creep velocity.

\[
S_{MIN} = \frac{\text{MD34040 $MA\_REFP\_VELO\_SEARCH\_MARKER}^2}{2 \times \text{MD32300 $MA\_MAX\_AX\_ACCEL}}
\]

Phase 1 is now complete. Reference point approach is continued with Phase 2.

Case 2: The machine axis is positioned on the reference cam
The machine axis remains at its starting position.
Phase 1 is now complete. Reference point approach is continued with Phase 2.

Case 3: The machine axis has no reference cam
Machine axes without reference cams remain at their starting position.
These include, for example:
- Machine axes that only have one zero mark along their entire traversing range
- Rotary axes that only have one zero mark per revolution
Zero must be entered in the following machine data for machine axes without a reference cam:
MD34000 $MA\_REFP\_CAM\_IS\_ACTIVE = 0 (axis with reference cam)

Figure 15-4 Minimum distance for deceleration
Phase 1 is now complete. Reference point approach is continued with Phase 2.

**Phase 1: Properties**

- Feed override active.
- Feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are active.
- The machine axis is stopped if the reference cam is not reached within the parameterized maximum distance:
  \[
  \text{MD34030 $MA\_REEP\_MAX\_CAM\_DIST$ \text{(max. distance to the reference cam)}}
  \]

**See also**

[Channelspecific referencing (Page 1152)]

**15.5.5 Phase 2: Synchronization with the zero mark**

**Phase 2: Graphic representation**

![Phase 2: Synchronization with the zero mark](image)

---

Figure 15-5  Phase 2: Synchronization with the zero mark
Phase 2: Start

Phase 2 is automatically started when Phase 1 has been completed without an alarm.

Initial situation:
The machine axis is positioned on the reference cam.

Zero mark search direction:
The direction of the zero mark search results from the settings in the machine data:

- MD34010 $MA_REFP_CAM_DIR_IS_MINUS (reference point approach in minus direction)
- MD34050 $MA_REFP_SEARCH_MARKER_REVERSE (direction reversal on reference cam)

Phase 2: Sequence

The synchronization in Phase 2 can be performed via the falling or rising edge of the reference cam. The parameterization is performed via:

MD34050 $MA_REFP_SEARCH_MARKER_REVERSE[<axis>] = <value>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Synchronization with falling reference cam edge</td>
</tr>
<tr>
<td>1</td>
<td>Synchronization with rising reference cam edge</td>
</tr>
</tbody>
</table>

Note
If the actual velocity of the machine axis at approach of the reference cam has not yet reached the target velocity of Phase 2 within the parameterized tolerance limits, Phase 1 will be restarted. This will be the case, for example, if the machine axis is positioned on the reference cam when reference point approach is started.

MD35150 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance)

Case 1: Synchronization with falling reference cam edge

During synchronization with falling reference cam edge, the machine axis accelerates to the parameterized reference point creep velocity opposite to the parameterized reference point approach direction (traversing direction of Phase 1)

After leaving the reference cam, the machine axis waits for the next encoder zero mark: DB31, ... DBX12.7 == 0

As soon as the encoder zero mark is detected, Phase 2 comes to an end. The machine axis continues at constant velocity and reference point approach is continued with Phase 3.

- MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)
15.5 Referencing with incremental measurement systems

Electronic reference cam shift

The electronic reference cam shift is used to compensate for expansions of the reference cam caused by temperature so that synchronization is always to the same encoder zero mark:
MD34092 $MA_\text{_REFP\_CAM\_SHIFT}$ (electronic reference cam shift for incremental measuring systems with equidistant zero marks)

With the electronic reference cam shift, synchronization is not performed immediately to the next encoder zero mark after detection of the reference cam edge, but only after the parameterized offset distance has been traversed.

Due to the determination of the distance traversed in the interpolator clock cycle since the detection of the reference cam edge, the effective shift distance is $s_{\text{shift}}$:

$$s_{\text{shift\_min}} = \text{MD34092 } $MA_\text{_REFP\_CAM\_SHIFT}$$
$$s_{\text{shift\_max}} = \text{MD34092 } $MA_\text{_REFP\_CAM\_SHIFT} + \text{MD34040 } $MA_\text{_REFP\_VELO\_SEARCH\_MARKER} \times \text{interpolator clock cycle}$$

The electronic reference cam shift acts in the direction of zero mark search.

![Diagram](image)

1. Reference cam shift

Figure 15-8  Electronic reference cam shift

**Requirement**

The electronic reference cam shift is only active for machine axes with reference cam:

MD34000 $MA_\text{_REFP\_CAM\_IS\_ACTIVE} == 1$

**Reference cam adjustment**

**Encoder with equidistant zero marks**

Always ensure that the reference cam of encoders that supply zero marks at equidistances is accurately adjusted so that the correct zero mark is always detected during reference point approach.

**Dynamic response**

The following factors influence the dynamic response from the arrival of the reference cam to the machine up to the detection of reference cam signals transferred from the PLC user program to the NC:

- Switching accuracy of the reference cam switch
- Delay of the reference cam switch (NC contact)
- Delay at the PLC input
- PLC cycle time
• Cycle time for updating the NC/PLC interface
• Interpolator clock cycle
• Position control cycle

**Notes on setting**

• Reference cam
  Aligning the signal edge of the reference cam directly between two zero marks has proven to be the most practical method.

• Electronic reference cam shift

<table>
<thead>
<tr>
<th><strong>WARNING</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Risk of collision</strong></td>
</tr>
<tr>
<td>If the reference cam adjustment is faulty or inaccurate, an incorrect zero mark can be evaluated. The controller then calculates an incorrect machine zero. As a result, the machine axis will approach the wrong positions. Software limit switches, protected areas and working area limitations act on incorrect positions and are therefore incapable of protecting the machine. The path difference is +/- of the path covered by the machine axis between two zero marks.</td>
</tr>
</tbody>
</table>

Information needed for parameterizing the electronic reference cam shift is to be found in the read-only machine data:

MD34093 $MA_REFP_CAM_MARKER_DIST (distance between reference cam/reference mark)

The indicated value is equivalent to the distance between departure from the reference cam and detection of the reference mark. If the values are too small, there is a risk that the determination of the reference point will be non-deterministic, due to temperature effects or fluctuations in the operating time of the cam signal.

**Phase 2: Properties**

• Feedrate override is **not** active.
  Machine axis moves internally when feedrate override = 100%.
  If a feedrate override of 0% is specified, an abort occurs.

• Feed stop (channel-specific and axis-specific) is active.

• NC stop and NC start are **not** active.

• If the machine axis does not arrive at Phase 2 within the parameterized distance of the reference mark (encoder zero mark), the machine axis will be stopped:
  MD34060 $MA_REFP_MAX_MARKER_DIST (max. distance to the reference mark)
15.5.6 Phase 3: Traversing to the reference point

Phase 3: Graphic representation

![Graph](image)

Figure 15-9 Phase 3: Traversing to the reference point

Phase 3: Start

At the end of Phase 2 the machine axis travels at reference point creep velocity. Therefore, as soon as Phase 2 is completed successfully without an alarm, Phase 3 is started without interruption.

**Initial situation**
The encoder zero mark has been detected.

Phase 3: Sequence

The machine axis moves at the assigned reference point positioning velocity:
MD34070 $MA_REFP_VELO_POS (reference point positioning velocity)
from the encoder zero mark detected in Phase 2 to the reference point.

The path $s_{ref}$ to be covered is calculated from the sum of the reference point distance plus reference point offset:

MD34080 $MA_REFP_MOVE_DIST (reference point distance)
MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)
When the reference point is reached, the machine axis is stopped and the actual-value system is synchronized with the reference point value \( n \) specified by the NC/PLC interface.

\[
\text{MD34100 $MA\_REFP\_SET\_POS[<n>]} \quad \text{(reference point value)}
\]

The selection of the reference point value is performed via the NC/PLC interface:

\[
DB31, \ldots \text{DBX2.4 ... 7} \quad \text{(reference point value 1 \ldots 4)}
\]

The actual-value system is synchronized to the reference point value that was selected at the time the reference cam was reached in Phase 1 (DB31, \ldots \text{DBX12.7} == 1).

The machine axis is now referenced. The interface signal is set as feedback to the PLC user program, depending on the active measuring system:

\[
DB31, \ldots \text{DBX60.4/5 (referenced/synchronized 1/2)} = 1
\]

**Features of Phase 3**

- Feed override active.
- Feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are active.
Special feature of Phase 3

In the following cases, the machine axis stops first after detection of the zero mark and then traverses back to the reference point:

- Because of the reference point positioning velocity, the sum of reference point distance and reference point offset is less than the required braking distance:
  \[ MD34080 + MD34090 < \text{"required braking distance due to MD34070"} \]
- The reference point is located, opposite to the current travel direction, "behind" the reference cam.

![Reference point distance plus reference point offset less than braking distance](image)

Figure 15-11 Reference point distance plus reference point offset less than braking distance

15.6 Referencing with distance-coded reference marks

15.6.1 General overview

Distance-coded reference marks

Measuring systems with distance-coded reference marks consist of two parallel scale tracks:

- Incremental grating
- Reference mark track

The distance between any two consecutive reference marks is defined. This makes it possible to determine the absolute position of the machine axis when two consecutive reference marks are crossed. For example, if the distance between the reference marks is approx. 10 mm, a traverse path of approx. 20 mm is all that is required to reference the machine axis.

Referencing can be performed from any axis position in the positive or negative direction (exception: end of travel range).
15.6.2 Basic parameter assignment

Linear measuring systems.

The following data must be set to parameterize linear measuring systems:

- The absolute offset between the machine zero point and the position of the first reference mark of the linear measuring system:
  MD34090 $MA_REFP_MOVE_DIST_CORR (reference point/absolute offset)
  See also below: Determining the absolute offset
- Orientation of the length measuring system (equidirectional or inverse) relative to the machine system coordinate system:
  MD34320 $MA_ENC_INVERS (length measuring system inverse to the machine system)

![DIADUR graduated glass scale with distance-coded reference marks](dimensions in mm for 20 mm scale division)

Rotary measuring system

For rotary measuring systems, the same applies as for linear measuring systems (see above).

Determining the absolute offset

The following procedure is recommended for the determination of the absolute offset between the machine zero point and the position of the first reference mark of a machine axis:

1. Enter the value zero for the absolute offset:
   MD34090 $MA_REFP_MOVE_DIST_CORR = 0
2. Perform reference point approach.
   Note: Reference point approach should be performed at a point in the machine where the exact position of the machine axis relative to machine zero can be determined easily with a laser interferometer, for example.
3. Determine the actual position of the machine axis via the operator interface screen.
4. Measure the current position of the machine axis with reference to the machine zero point.
5. Calculate absolute offset and enter in MD34090.
The absolute offset is calculated with respect to the machine coordinate system and depending on the orientation of the measuring system (equidirectional or inverse) as:

<table>
<thead>
<tr>
<th>Orientation of the measuring system</th>
<th>Absolute offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equidirectional</td>
<td>Measured position + displayed actual position</td>
</tr>
<tr>
<td>Opposite direction</td>
<td>Measured position - displayed actual position</td>
</tr>
</tbody>
</table>

⚠️ WARNING

Reference point deviation

After determining the absolute offset and the entry in MD34090, the reference point traversing for the machine axis must be carried out once more.

Referencing methods

Referencing with distance-coded reference marks can be performed in one of two ways:

- **Evaluation of two consecutive reference marks:**
  MD34200 $MA\_ENC\_REFP\_MODE (referencing mode) = 3
  Advantage:
  - Short travel path

- **Evaluation of four consecutive reference marks:**
  MD34200 $MA\_ENC\_REFP\_MODE = 8
  Advantage:
  - Plausibility check by NC is possible
  - Increase in reliability of referencing result
15.6.3 Time sequence

**Time sequence**

Referencing with distance-coded reference marks can be divided into two phases:

- **Phase 1:** Travel across the reference marks with synchronization
- **Phase 2:** Travel to a fixed destination point

![Diagram of Time sequence](chart.html)

**Figure 15-13** Distance-coded reference marks

15.6.4 Phase 1: Travel across the reference marks with synchronization

**Phase 1: Start**

To start the reference point approach, see Sections "Axis-specific referencing (Page 1150)" and "Channelspecific referencing (Page 1152)".

**Reference cam**

In measuring systems with distance-coded reference marks, reference cams are not required for the actual referencing action. For functional reasons, however, a reference cam is required for channel-specific reference point approach and reference point approach from the part program (G74) before the traversing range end of the machine axis.
Phase 1: Sequence

**Sequence without contact with a reference cam**

Once the reference point approaching process is started, the machine axis accelerates to the reference point shutdown speed set by means of parameter assignment:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

Once the number of reference marks set by means of parameter assignment has been crossed, the machine axis is stopped again and the actual value system of the machine axis is synchronized to the absolute position calculated by the NC.

**Sequence when starting from the reference cam**

If the machine axis is at the reference cam at the start of the reference point traversing, it accelerates to the parameterized reference point creep velocity against the parameterized reference point approach direction:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

MD34010 $MA_CAM_DIR_IS_MINUS (reference point approach in minus direction)

That ensures that the machine axis does not reach the travel range limit before it has crossed the parameterized number of reference marks.

Once the number of reference marks set by means of parameter assignment has been crossed, the machine axis is stopped again and the actual value system of the machine axis is synchronized to the absolute position calculated by the NC.

**Sequence when contact is made with reference cam during referencing**

Once the reference point approaching process is started, the machine axis accelerates to the reference point shutdown speed set by means of parameter assignment:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reference point creep velocity)

Before the machine axis travels over the parameterized number of reference marks, it makes contact with the reference cam. It is then reversed and reference mark search is restarted in the opposite direction.

Once the number of reference marks set by means of parameter assignment has been crossed, the machine axis is stopped again and the actual value system of the machine axis is synchronized to the absolute position calculated by the NC.

**Plausibility check of the reference mark distance**

An error occurs if, during reference point traversing for two subsequent reference marks, the NC determines a distance greater than twice the parameterized reference mark distance.

MD34300 $MA_ENC_REFP_MARKER_DIST (reference mark distance)

The machine axis will then traverse in opposite direction at half the parameterized reference point creep velocity (MD34040) and the search for reference mark is restarted.

If a faulty reference mark distance is detected again, the machine axis is stopped and the reference point traversing is aborted (alarm 20003 “fault in the measuring system”).
Abort criterion

If the parameterized number of reference marks is not detected within the parameterized distance, the machine axis is stopped and reference point traversing is aborted.

MD34060 $MA_REFP_MAX_MARKER_DIST (max. distance to the reference mark)

Features of Phase 1

After Phase 1 is successfully completed, the actual value system of the machine axis is synchronized.

15.6.5 Phase 2: Traversing to the target point

Phase 2: Start

Phase 2 is automatically started when Phase 1 has been completed without an alarm.

Initial situation:

- The machine axis is positioned directly behind the last of the parameterized number of reference marks.
- The actual value system of the machine axis is synchronized.

Phase 2: Sequence

In Phase 2, the machine axis completes reference point approach by traversing to a defined target position (reference point). This action can be suppressed in order to shorten the reference point approach:

MD34330 $MA_STOP_AT_ABS_MARKER = <value>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Travel to target position</td>
</tr>
<tr>
<td>1</td>
<td>No travel to target position</td>
</tr>
</tbody>
</table>

Travel to target position (normal case)

The machine axis accelerates to the parameterized reference point position velocity and travels to the parameterized target point (reference point):

MD34070 $MA_REFP_VEL_POS (reference point positioning velocity)

MD34100 $MA_REFP_SET_POS (reference point value)

The machine axis is referenced. To identify this, the NC sets an interface signal for the measuring system that is currently active:

DB31, ... DBX60.4/60.5 (referenced/synchronized 1/2) = 1
No travel to target position
The machine axis is now referenced. To identify this, the NC sets an interface signal for the measuring system that is currently active:
DB31, ... DBX60.4/60.5 (referenced/synchronized 1/2) = 1

Features of Phase 2
Phase 2 will display different characteristics, depending on whether a reference cam is parameterized for the machine axis.

Machine axis without reference cam
MD34000 $MA_REFP_CAM_IS_ACTIVE (axis with reference cam) = 0
Properties:
- Feed override active.
- The feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are active.

Machine axis with reference cam
MD34000 $MA_REFP_CAM_IS_ACTIVE (axis with reference cam) = 1
Properties:
- Feedrate override is not active. 
  Machine axis moves internally when feedrate override = 100%.
  If a feedrate override of 0% is specified, an abort occurs.
- The feed stop (channel-specific and axis-specific) is active.
- NC stop and NC start are not active.
- If the parameterized number of reference marks is not detected within the parameterized distance after the exit of the reference cam, the machine axis will be stopped.
  MD34060 $MA_REFP_MAX_MARKER_DIST (max. distance to the reference mark)

Special features of rotary measuring systems
On rotary distance-coded measuring systems, the absolute position can only be determined uniquely within one revolution. Depending on the mechanical mounting of the encoder, the overtravel of the absolute position in the hardware does not always coincide with the traversing range of the rotary axis.

Special features of modulo rotary axes
With modulo rotary axes, the reference point position is mapped on the parameterized modulo range:
MD30330 $MA_MODULO_RANGE (size of the modulo range)
15.7 Referencing by means of actual value adjustment

15.7.1 Actual value adjustment to the referencing measurement system

**Function**

When actual value adjustment to the referencing measuring system is performed, the resulting absolute actual position after successful referencing of the measuring system of a machine axis is transferred directly to all other measuring systems of the machine axis, and the machine axis is designated as referenced:

DB31, ... DBB60.4 / 60.5 (referenced/synchronized 1/2) = 1

**Advantage**

When the machine axis switches from an explicitly referenced measuring system to the measuring system referenced by actual value adjustment, continuous servo control is assured (servo enable active) because the matched actual position prevents a sudden change in actual value.

**Note**

In order to improve positioning precision by determining the measuring-system-specific encoder fine information, we recommend explicitly re-referencing the measuring system previously referenced by actual value adjustment after switching over.

**Activation**

The activation of the actual value adjustment to the referencing measuring system is machine-specifically carried out via:

MD34102 $MA_REFP_SYNC_ENCS = 1
15.7.2 Actual value adjustment for measuring systems with distance-coded reference marks

Function

In order to improve positioning precision by determining the measuring-system-specific encoder fine information, we recommend explicitly re-referencing the measuring system previously referenced by actual value adjustment after switching over the measuring system.

If an encoder with distance-coded reference marks is used for the passive measuring system, referencing can be avoided if the following conditions are met:

1. Active measuring system: indirect measuring system (motor measuring system) with absolute encoder, for example
2. Passive measuring system: Direct measuring system with distance-coded reference marks
3. Travel movement of the machine axis with the referenced indirect measuring system before measuring system switchover in which the number of reference marks required for referencing are crossed. This automatically references the passive direct measuring system.

Parameterization

In addition to the specific machine data required to reference the individual measuring systems, the following machine data must be set:

- Enable actual value adjustment:
  MD34102 $MA_REFP_SYNC_ENCS = 1

- Direct measuring system with distance-coded reference marks:
  - MD34200 $MA_ENC_REFP_MODE[measuring system] = 3
    Distance-coded reference marks
  - MD30242 $MA_ENC_IS_INDEPENDENT[measuring system] = 2

During actual value adjustment, the passive direct measuring system is adjusted to the actual position of the active indirect measuring system, but is not marked as referenced. After the parameterized number of reference marks have been crossed, the passive direct measuring system is automatically referenced. Referencing is performed in every operating mode.
Sequence

1. Initial situation: Neither of the measuring systems are referenced:
   DB31, ... DBX60.4 = 0 (referenced / synchronized 1)
   DB31, ... DBX60.5 = 0 (referenced / synchronized 2)

2. Reference the indirect measuring system according to the measuring system type:
   DB31, ... DBX60.4 = 1 (referenced / synchronized 1)
   DB31, ... DBX60.5 = 0 (referenced / synchronized 2)

3. Traverse the machine axis across the parameterized number of reference marks.
   This automatically references the direct measuring system:
   DB31, ... DBX60.4 = 1 (referenced / synchronized 1)
   DB31, ... DBX60.5 = 1 (referenced / synchronized 2)

15.8 Referencing in follow-up mode

Function

Incremental measuring systems and measuring systems with distance-coded reference marks can be referenced even when the machine axis is in follow-up mode. Prerequisite for this is the correct parameterization of the reference point approach according to the used measuring system (see Section "Referencing with incremental measurement systems (Page 1154)" and "Referencing with distance-coded reference marks (Page 1167)").

When referencing in follow-up mode the machine axis is moved not by the NC but by means of an external travel motion over the encoder zero mark and the parameterized number of distance-coded reference marks. The measuring system is referenced when the encoder zero mark or parameterized number of distance-coded reference marks are detected.

Note

Reproducibility of the referencing result

In NC-guided reference point approach, reproducibility of the referencing result is ensured through adherence to the assigned traverse velocities during the referencing operation. During referencing in follow-up mode, responsibility for achieving reproducibility of the referencing results lies with the machine manufacturer / user.

Unique zero mark

Referencing of an incremental measuring system is based on the explicit position of the encoder zero mark relative to the overall traversing range of the machine axis.

Because the reference cam signal is not evaluated by the NC during referencing in follow-up mode, unique identification of the reference point when referencing in follow-up mode will only result with:

- Only one encoder zero mark in the traversing range of the machine axis.
- Linear measuring systems with distance-coded reference marks.
- Modulo rotary axes (absolute position within one revolution).
Zero mark selection when several zero mark signals occur

If several encoder zero marks are detected in the traversing range of the machine axis due to machine-specific factors, e.g. reduction gear between encoder and load, a BERO must be mounted on the machine and connected via a digital input of the NCU interface in order to clearly determine the reference point.

Note
BERO signal: Zero mark selection
The processing of the BERO signal is performed exclusively in the drive. Connection and parameterization, see Section "Hardware signals (Page 1154)".

Enable

The "Referencing in follow-up mode" function is enabled with:

MD34104 $MA_REFP_PERMITTED_IN_FOLLOWUP = TRUE

Starting the referencing operation

If the machine axis is in follow-up mode (DB31, ... DBX61.3 == TRUE) at the start of reference point approach, the measuring system is referenced in follow-up mode.

If the machine axis is not operating in the follow-up mode at the start of reference point traversing, the "normal" from the NC-controlled reference point travels is carried out.

Referencing in follow-up mode can be started in the following modes:

- JOG-REF: Traversing keys
- AUTOMATIC: Part program command G74

Sequence of the referencing operation (JOG-REF mode)

1. Activate follow-up mode of machine axis:
   DB31, ... DBX1.4 (follow-up mode) = 0
   DB31, ... DBX2.1 (controller enable) = 0
2. Take into account activation of follow-up mode:
   DB31, ... DBX61.3 (follow-up active) = 1
3. Switch to JOG-REF mode.
4. External motion of machine axis across encoder zero mark or parameterized number of distance-coded reference marks. The referencing operation is started internally in the NC as soon as the machine axis is moved:
   DB31, ... DBX61.4 (axis/spindle stationary) = 0
5. The measuring system is referenced after the encoder zero mark or the assigned number of distance-coded reference marks have been successfully detected:
   DB31, ... DBX60.4/60.5 (referenced/synchronized 1/2) = 1

Aborting the reference operation
An active referencing operation can be aborted by:

- Deselecting follow-up mode
- NCK Reset

Response when measuring systems are already referenced

A measuring system that has already been referenced can only be re-referenced in AUTOMATIC mode using part program statement G74.

Sequence of referencing operation (AUTOMATIC mode)

1. Switch to AUTOMATIC mode.
2. Start the part program.
3. Activate follow-up mode of machine axis:
   DB31, ... DBX1.4 (follow-up mode) = 0
   DB31, ... DBX2.1 (controller enable) = 0
4. Take into account activation of follow-up mode:
   DB31, ... DBX61.3 (follow-up active) = 1
5. The referencing operation is started internally in the NC as soon as part program statement G74 is processed.
6. External motion of machine axis across encoder zero mark or parameterized number of distance-coded reference marks.
7. The measuring system is referenced after the encoder zero mark or the assigned number of distance-coded reference marks have been successfully detected:
   DB31, ... DBX60.4/60.5 (referenced/synchronized 1/2) = 1
8. The block change occurs after the referencing operation has been successfully completed.

Aborting the reference operation

An active referencing operation can be aborted by:

- Deselecting follow-up mode
- NCK Reset

Response when measuring systems are already referenced

A measuring system that you have already referenced can be re-referenced.
15.9 Referencing with absolute encoders

15.9.1 Information about the adjustment

Machine axes with absolute encoder

The advantage of machine axes with absolute encoder is that after a one time adjustment procedure, the necessary reference point traversing with incremental measuring systems (e.g. build-up of control, de-selection of "Parking" of machine axes etc.) can be skipped and the actual value system of the machine axis can be immediately synchronized to the determined absolute position.

Adjustment

Adjustment of an absolute encoder involves matching the actual value of the encoder with the machine zero once and then setting it to valid.

The current adjustment status of an absolute encoder is displayed in the following axis-specific machine data of the machine axis, to which it is connected:

MD34210 $MA_ENC_REFP_STATE (status of absolute encoder)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Encoder not calibrated</td>
</tr>
<tr>
<td>1</td>
<td>Encoder adjustment enabled</td>
</tr>
<tr>
<td>2</td>
<td>Encoder is calibrated</td>
</tr>
</tbody>
</table>

Adjustment methods

The following adjustment methods are supported:

- Adjustment by entering a reference point offset
- Adjustment by entering a reference point value
- Automatic adjustment with probe
- Adjustment with BERO

Readjustment

Readjustment of the absolute encoder is required after:

- Gear change between load and absolute encoder
- Removal/installation of the absolute encoder
- Removal/installation of the motor with the absolute encoder
- Data loss in the static NC memory
Battery failure

Setting actual value (PRESETON)

### WARNING

#### Data backup

During the back-up of machine data of a machine A, the encoder status of the machine axis (MD34210) is also backed up.

During loading of this data record into a machine B of the same type, e.g. in the context of a serial start-up or after a case of maintenance, the referenced machine axes will be automatically regarded as adjusted / referenced by the NC. It is the special responsibility of the machine manufacturer / user to undertake a readjustment in such cases.

See also explanations regarding machine data:

MD30250 $MA_ACT_POS_ABS (absolute encoder position at the time of switch-off)

### Note

The controller can detect a required readjustment of the absolute encoder only during the following events:

- Gear change with change of gear ratio
- Addressing the zero-mark monitoring
- New encoder serial number after change of the absolute encoder

The controller then sets the status of the absolute encoder to "0":

MD34210 $MA_ENC_REFP_STATE = 0 (encoder not adjusted)

The following alarm is displayed:

Alarm 25022 "Axis <axis name> encoder <number> warning 0"

If the zero-mark monitoring responds, the following alarm is also displayed:

Alarm 25020 "Axis <axis name> zero-mark monitoring of active encoder"

In all other cases (e.g. PRESETON) it is the sole responsibility of the user to display the misalignment of the absolute encoder by manually setting the status to "0" and to carry out a readjustment.

### 15.9.2 Calibration by entering a reference point offset

#### Function

During adjustment by entering the reference point offset, the difference between the position displayed on the operator interface and the true actual position in the machine is determined and made known to the NC as reference point offset.
Procedure

1. Determining the position of the machine axis with reference to the machine zero point via e.g.:
   - position measurement (e.g. laser interferometer)
   - Moving the machine axis to a known position (e.g., fixed stop)
2. Reading the displayed actual position of the machine axis on the operator interface.
3. Calculating the reference point offset (difference between the actual positions determined under point 1 and 2) and entering in machine data:
   MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)
4. Marking the absolute value encoder as adjusted:
   MD34210 $MA_ENC_REFP_STATE = 2

   Note
   The encoder adjustment does not become active until the next time the encoder is activated (e.g., when the controller is powered up).

5. Initiate POWER ON reset.
6. Controlling the position of the machine axis displayed on the operator interface.

   Note
   Backlash compensation
   If backlash compensation is parameterized for a measuring system with absolute value encoder, the following must be observed:
   No backlash is permitted during machine axis travel to the adjusted machine position.

   Activate reference point offset permanently
   The entered reference point offset (MD34090) will be permanently active only after initial POWER ON - Reset. If the machine axis is moved after the absolute encoder adjustment without an interim POWER ON - Reset, the reference point offset entered in the machine data can be overwritten, for example, as part of internal overrun offset.

   Checking the actual position
   Following adjustment of the absolute encoder, we recommend that you verify the actual position of the machine axis the next time you power up the controller (POWER ON).

15.9.3 Adjustment by entering a reference point value

Function

During adjustment by entering the reference point value, the absolute position of the machine axis with reference to the machine zero point is determined by e.g.:

- Position measurement (e.g. laser interferometer)
- Moving the machine axis to a known position (e.g. fixed stop)
This determined position value will be made known to the NC as the reference point value. The NC then calculates the reference point offset from the difference between the encoder absolute value and the reference point value.

**Procedure**

1. Check whether the referencing mode is set to "Assume the reference point value":
   
   \[
   \text{MD34200 $MA\_ENC\_REFP\_MODE == 0}
   \]
   
   If not, enter the value 0 in the machine and trigger a Power On reset.

2. Traverse the machine axis in JOG mode to the position to be measured (e.g. with a laser interferometer) or to a known position (e.g. fixed stop).

   **Note**

   The machine axis can only be traversed in the direction enabled for referencing with the travel keys:

   \[
   \text{MD34010 $MA\_REFP\_CAM\_DIR\_IS\_MINUS (approach reference point in minus direction)}
   \]

   To avoid an invalid position because of backlash in the drive train, the known position must be approached at low velocity.

3. Enter the position of the machine axis relative to machine zero as the reference point value in the machine data:

   \[
   \text{MD34100 $MA\_REFP\_SET\_POS = Position}
   \]

4. Enable encoder adjustment:

   \[
   \text{MD34210 $MA\_ENC\_REFP\_STATE = 1}
   \]

5. Trigger a Power On reset for acceptance of the entered machine data values.

6. Switch to JOG-REF mode.
7. Operate the travel key used for referencing in step 2.  
The machine axis does not move when the traversing key is actuated!  
The NC calculates the reference point offset from the entered reference point value and  
that given by the absolute encoder. The result is entered into the machine data:  
MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)  
The status of the absolute encoder is set to "Encoder is adjusted":  
MD34210 $MA_ENC_REFP_STATE = 2  
The actual value system of the machine axis is synchronized.  
The machine axis is now referenced. As identification, the NC sets the appropriate interface  
signal based on which measuring system is currently active:  
DB31, ... DBB60.4 / 60.5 (referenced/synchronized 1 / 2) = 1  

8. Initiate Power On reset.  

**Note**  
*Activate reference point offset permanently*  
The entered reference point offset (MD34090) will only be permanently active after Power  
On reset.  

If the machine axis is moved after the absolute encoder adjustment without an interim  
Power On reset, the reference point offset entered in the machine data can be overwritten,  
for example, within internal overrun corrections.  

**Checking the actual position**  
Following adjustment of the absolute encoder, we recommend that you verify the actual  
position of the machine axis the next time you power up the controller (Power On).  

---  

**15.9.4 Automatic calibration with probe**  

**Function**  
In automatic adjustment with a probe, a known position in the machine is approached with the  
machine axis from a part program. The position value is stored in the NC as a reference point  
value. The position is reached when the probe switches, and the NC then calculates the  
reference point offset from the difference between the encoder value and reference point value.  

**Note**  
*Part program for automatic adjustment*  
The part program for automatic adjustment using a probe must be created by the machine  
manufacturer / user for the specific requirements of the machine.  

**Freedom from collision**  
Because actual-value-related monitoring is not active for the machine axes being referenced,  
the machine operator must take special care to ensure that collisions do not occur in the  
machine while the machine axes are being moved!
Part program
The part program for automatic adjustment of absolute encoders with probe must perform the points listed below for each axis in the order indicated:

1. Approach the adjustment position of machine axis, which is detected from the probe response.
   The position must be approached several times from the same direction, but at a velocity which is gradually reduced on each approach, to ensure that the measured value obtained is as accurate as possible. The measured value is stored in system variable $AA_IM.

2. Calculating and writing the reference point offset:

   MD34090 $MA_REFP_MOVE_DIST_CORR = MD34100 $MA_REFP_SET_POS – $AA_IM

3. Set the absolute encoder status to “Encoder is adjusted”:
   MD34210 $MA_ENC_REFP_STATE = 2

Sequence
Proceed as follows for automatic adjustment with probe:

1. Enable part program start even for non-referenced machine axes:
   MD20700 $MC_REFP_NC_START_LOCK = 0

2. Enter the machine axis position relative to machine zero when probe is switched as the reference point value for all relevant machine axes:
   MD34100 $MA_REFP_SET_POS = reference point value

3. Activate NCK-Reset for the acceptance of the entered machine data values.

4. Start part program.

5. After completion of the part program, re-secure the partial program start for machine axes which are not referenced:
   MD20700 $MC_REFP_NC_START_LOCK = 1

6. Initiate POWER ON - Reset so that the reference point offset written by the part program is permanently active:
   MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset)

Note
Activate reference point offset permanently
The entered reference point offset (MD34090) will only be permanently active after POWER ON - Reset.

If the machine axis is moved after the absolute encoder adjustment without an interim POWER ON - Reset, the reference point offset entered in the machine data can be overwritten, for example, as part of internal overrun offset.

Checking the actual position
Following adjustment of the absolute encoder, we recommend that you verify the actual position of the machine axis the next time you power up the controller (POWER ON).
15.9.5 Adjustment with BERO

Function

For adjustment using BERO, a reference point approach to a defined machine position is performed the same as for incremental measuring systems. In this case the BERO replaces the encoder zero mark that the absolute encoder does not have. After successful completion of reference point approach, the NC automatically calculates the reference point offset from the difference between the encoder absolute value and the parameterized reference point value.

Parameterization

NC: Referencing mode
The referencing mode should be set to "Referencing of incremental, rotary or linear measuring systems: Zero pulse on the encoder track":

MD34200 $MA_ENC_REFP_MODE[<axis>] = 1

NC: Reference point value
The reference point value is parameterized via:

MD34100 $MA_REFP_SET_POS[<axis>] = <reference point value>

Drive: Equivalent zero mark
The digital input on the NCU interface to which the BERO is connected must be set in parameter p0494 or p0495.

Execution

Reference point approach can be started manually in JOG-REF mode or in AUTOMATIC or MDA mode via a part program (G74).

After successful completion of the reference point approach, the absolute encoder is adjusted and the actual-value system of the machine axis is synchronized.

As feedback for the PLC user program, the NC sets the NC/PLC interface signal for the machine axis, depending on the active measuring system:

DB31, ... DBB60.4/60.5 (referenced/synchronized 1/2) = 1

Note

If the BERO is removed after adjustment of the absolute encoder, the referencing mode must be re-parameterized to "Referencing with absolute encoder".

MD34200 $MA_ENC_REFP_MODE[<axis>] = 0
15.9.6 Reference point approach with absolute encoders

Traversing movement release

If for a machine axis with adjusted absolute value encoder as active measuring system, reference point traversing is activated (manual in the mode JOG-REF or automatic according to part program instruction \texttt{G74}), the machine axis travels depending on the parameterized release traversing movement.

\[ \text{MD34330 } $MA\_REFP\_STOP\_AT\_ABS\_MARKER = \langle \text{Value} \rangle \]

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Traversing is enabled. When reference point approach is initiated, the machine axis moves to the reference point position. Reference point approach is completed when the reference point position is reached.</td>
</tr>
<tr>
<td>1</td>
<td>Traversing is not enabled. After the activation of the reference point travel, the machine axis does not travel and the reference point travel is immediately completed.</td>
</tr>
</tbody>
</table>

15.9.7 Reference point approach in rotary absolute encoders with external zero mark

Function

To be able to use the reference point approach with a zero mark, as is usual in incremental encoders (refer to Section "Referencing with incremental measurement systems (Page 1154)"), also with absolute encoders, the missing HW zero marks are created in software form once every encoder revolution, always at the same position within the rotation.

Difference compared to referencing with incremental encoders

An absolute encoder with replacement zero mark should not be considered as a complete equivalent of an incremental encoder. All the properties of the absolute encoder are retained. The following table lists the different properties of incremental and absolute encoders:

| Table 15-1 Properties of incremental and absolute encoders |
|-----------------|-----------------|
| Property        | Incremental encoder | Absolute encoder |
| Encoder type    | MD30240 $MA\_ENC\_TYPE = 1 | $MA\_ENC\_TYPE = 4 |
| Internal encoder position | MD30250 $MA\_ACT\_POS\_ABS | Value is updated only in MD34210 $MA\_REFP\_MOVE\_DIST\_CORR |
| Traversing range extension | MD30270 $MA\_ENC\_ABS\_BUFFERING | Value is updated only in MD30270 $MA\_ENC\_ABS\_BUFFERING = 0 |
| Reference point offset | MD34090 $MA\_REFP\_MOVE\_DIST\_CORR | Value input allowed Value is updated exclusively via control |

R1: Referencing

15.9 Referencing with absolute encoders
### R1: Referencing

#### 15.9 Referencing with absolute encoders

<table>
<thead>
<tr>
<th>Property</th>
<th>Incremental encoder</th>
<th>Absolute encoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported referencing types</td>
<td>MD34200 $MA_ENC_REFP_MODE</td>
<td>$MA_ENC_REFP_MODE = 1, 2, 3, 4, 5, 6, 7</td>
</tr>
<tr>
<td>Adjustment status</td>
<td>MD34210 $MA_ENC_REFP_STATE = 0, 1, 2</td>
<td>Automatic encoder misalignment during shut down while in motion.</td>
</tr>
<tr>
<td>Absolute position modulo range</td>
<td>MD34220 $MA_ENC_ABS_TURNS_MODULO</td>
<td>$MA_ENC_ABS_TURNS_MODULO = 0</td>
</tr>
<tr>
<td>Encoder serial no.</td>
<td>MD34230 $MA_ENC_SERIAL_NUMBER</td>
<td>$MA_ENC_SERIAL_NUMBER = 0</td>
</tr>
<tr>
<td>Transfer of series startup files</td>
<td>Without any restrictions.</td>
<td>Due to encoder properties MD30250, MD30270, MD34090, MD34210, MD34220, MD34230 only possible with limitations.</td>
</tr>
<tr>
<td>Activation time</td>
<td>0 seconds</td>
<td>several seconds</td>
</tr>
<tr>
<td>Zero mark</td>
<td>1 per encoder revolution</td>
<td>None</td>
</tr>
<tr>
<td>Zero-mark monitoring</td>
<td>Hardware</td>
<td>Software</td>
</tr>
<tr>
<td>Position after POWER ON without actual value buffering</td>
<td>0.0</td>
<td>Last position within MD34220.</td>
</tr>
<tr>
<td></td>
<td>MD34210 = 0</td>
<td>MD30270 = 1</td>
</tr>
<tr>
<td>Position after POWER ON with actual value buffering</td>
<td>Last standstill position before deactivation.</td>
<td>Last position including small movements during POWER OFF.</td>
</tr>
<tr>
<td></td>
<td>MD34210 = 1</td>
<td>MD30270 = 0</td>
</tr>
<tr>
<td>Referenced after POWER ON</td>
<td>depends on adjustment status</td>
<td></td>
</tr>
</tbody>
</table>

#### Requirement

The function can be used only with rotary absolute encoders:

- MD31000 $MA_ENC_IS_LINEAR = 0
- MD30240 $MA_ENC_TYPE = 4

#### Parameterization

- reference point approach with zero marks:
  MD34200 $MA_ENC_REFP_MODE = 1
- A reference point offset should not be input in the following MD:
  MD34090 $MA_REFP_MOVE_DIST_CORR
  This MD describes, in connection with absolute encoders, the offset between machine and absolute encoder zero, and it therefore has a different meaning.
• The load-side zero mark search rate MD34040 $MA_REFP_VELO_SEARCH_MARKER should not exceed the limiting frequency of the absolute trace of the encoder MD36302 $MA_ENC_FREQ_LIMIT_LOW.

  If the speed is too high, absolute information cannot be read any more, and thus, no equivalent zero marks are generated.

• If no zero marks are found within:
  MD34060 $MA_REFP_MAX_MARKER_DIST
  otherwise, an alarm will be triggered.

• A start of the zero mark search with the override of a BERO (MD34200 = 5) is not supported. MD34200 = 0 is to be used as a equivalent.

• The following MD must be set if the absolute encoder retains even the referenced status through POWER OFF, besides the last position:
  MD34210 $MA_ENC_REFP_STATE = 2

### Data backup and standard commissioning

Some properties of an absolute encoder restrict the transfer of series startup files to other machines. The following machine data must be checked and corrected if necessary after loading the series startup:

- MD30250 $MA_ACT_POS_ABS (internal encoder position)
- MD30270 $MA_ENC_ABS_BUFFERING (traversing range extension)
- MD34090 $MA_REFP_MOVE_DIST_CORR (absolute offset)
- MD34210 $MA_ENC_REFP_STATE (adjustment status)
- MD34220 $MA_ENC_ABS_TURNS_MODULO (Modulo range)
- MD34230 $MA_ENC_SERIAL_NUMBER (encoder serial number)

### 15.9.8 Enabling the measurement system

The measuring system of a machine axis is activated in the following cases:

- Power up of the control (POWER ON)
- Activation of the measuring system via interface signal (deselection of "parking"): DB31, DBB1.5 / 1.6 (position measuring system 1/2)
  DB31, DBB2.1 (servo enable)
- Violation of the assigned encoder limit frequency (spindles):
  MD36300 $MA_ENC_FREQ_LIMIT

When the measuring system is activated, the NC synchronizes the actual value system of the machine axis with the current absolute value. Traversing is disabled during synchronization for axes but not for spindles.
**Parameterizing the encoder limit frequency (spindles)**

The EQN 1325 absolute encoder made by Heidenhain has an incremental track and an absolute track.

If a spindle is driven at a speed above the encoder limit frequency of the incremental track, the substantially lower limit frequency of absolute track must be parameterized as the encoder limit frequency.

**MD36300 $MA_ENC_FREQ_LIMIT**

Otherwise an incorrect absolute position would be read because the parameterized encoder limit frequency is not reached when the measuring system is activated. This would cause a position offset in the actual value system of the machine axis.

**Determining the encoder limit frequency**

The encoder limit frequency to be parameterized is derived from the smaller of the two following limit speeds:

- **Encoder**
  The limit speed or encoder limit frequency is listed in the data sheet of the encoder (e.g., limit speed = 2000 [rpm])

- **NC**
  Due to the NC-internal evaluation process, the maximum limit speed for which error-free calculation of the absolute value by the NC is possible is 4 encoder revolutions per interpolator clock cycle.
  
  For an interpolator clock cycle of, for example, 12 ms: Limit speed = 4 / 12 ms = 20,000 rpm
  
  The limiting frequency corresponding to the limiting speed is calculated to be:

  \[
  \text{MD36300} = \frac{4 \times \text{MD31020}}{\text{MD10050} \times \text{MD10070}}
  \]

**MD31020 $MA_ENC_RESOL** (encoder pulses per revolution)

**MD10050 $MN_SYSCLOCK_CYCLE_TIME** (basic system clock cycle)

**MD10070 $MN_IPO_SYSCLOCK_TIME_RATIO** (factor for the interpolation cycle)

**Note**

The position control switching speed relevant for spindles is set according to the encoder limiting frequency of the absolute value encoder of the spindle:

**MD35300 $MA_SPIND_POSCTRL_VELO** (position control switching speed)

**MD36300 $MA_ENC_FREQ_LIMIT** (Encoder limit frequency)
15.9.9 Referencing variants not supported

The following referencing variants are not supported when used with absolute encoders:

- Referencing/calibrating with encoder zero mark
- Distance-coded reference marks
- BERO with two-edge evaluation

15.10 Automatic restoration of the machine reference

Without a defined machine reference when traversing machine axes, no position-dependent functions such as transformations or tool frames can be executed. In various machine situations, these functions must be available immediately with the encoder activation, e.g. after the control is switched on or after terminating "Parking a machine axis (Page 113)" to traverse the axes. However, the machine axes should not or cannot be traversed again for referencing.

Absolute encoders

For measuring systems with adjusted absolute encoders, the machine reference is restored immediately without any additional measures when the encoder value is read.

Incremental encoders

With incremental measuring systems, the machine reference can be restored without traversing the axes through "Automatic referencing" or "Restoration of the actual position".

Supplementary conditions

<table>
<thead>
<tr>
<th>WARNING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect synchronization of the position measuring system caused by offset of the actual machine axis position</td>
</tr>
</tbody>
</table>

During the time in which the measuring system of the machine axis is switched off, it is not permissible that the axis is mechanically moved. Otherwise this results in an offset between the last buffered actual position and the real actual position of the machine axis. This would lead to an incorrect synchronization of the measuring system resulting in danger to personnel and machine.

The machine manufacturer must provide such measures as holding brakes, etc. on the machine so that the actual position is not changed, and this must be ensured by the user. The responsibility for this rests exclusively with the machine manufacturer / user.

If axis motion cannot be prevented mechanically in the shutdown state, either an absolute encoder must be used or the axis must be referenced again with reference point approach after switching on.
Note
SMExx sensor modules
Automatic referencing or restoration of the actual position to the last buffered position after
restarting the control is only possible in conjunction with SMExx (externally mounted) sensor
modules. When using SMCxx (cabinet) or SMIxx (integrated) sensor modules, the actual
position cannot be restored after restarting the control (power on). The measuring system of
the machine axis must be referenced again.

15.10.1 Automatic referencing

Function
During automatic referencing, the actual position of the active measuring system of the
machine axis is set to the last buffered position and "referenced" set as encoder state after
switching on the control. This makes it possible to start programs in the AUTOMATIC and MDI
modes directly after run-up of the control.

Requirements
- The active measuring systems when the control is switched on must already have been
  referenced once before switching off.
- At the time the control is switched off, the machine axis must be at standstill with "Exact
  stop fine" (DB31, ... DBX60.7 == 1) are located.

Note
If the machine axis is not at standstill with "Exact stop fine" when switching off, the actual
position will be initialized with " 0" when switching on. "Not referenced" is displayed as the
encoder state.

Parameter assignment
The automatic referencing is enabled by setting the encoder state to "Automatic referencing
is enabled, but the encoder has not been referenced":

$MD34210 \text{ $MA\_ENC\_REFP\_STATE}<\text{encoder}> = 1$

After the measuring system has been referenced, the encoder state displays that automatic
referencing will be executed the next time the encoder is activated:

$MD34210 \text{ $MA\_ENC\_REFP\_STATE}<\text{encoder}> == 2$

NC/PLC interface signals
After automatic referencing, the encoder state "Referenced" is displayed for the active
measuring system:

$DB31, ... DBX60.4/5 == 1$ (referenced/synchronized 1/2)
Supplementary conditions

Encoder activation with MD34210 $MA_ENC_REFP_STATE[<encoder>] == 1

An encoder state equal to "1" at the time of the encoder activation means that "Automatic referencing" has been enabled. However, the measuring system has either not been referenced yet or the machine axis was not switched off at standstill in the "Exact stop fine" state. The following is set for the machine axis or the active measuring system:

- Actual position = 0
- Active measuring system, encoder state = "Not referenced":
  DB31, ... DBX60.4 / .5 = 0 (referenced/synchronized 1/2)

References

Description of Functions, Basic functions, Section "R1 Referencing" > "Referencing with incremental measurement systems" (Page 1154) > "

15.10.2 Restoration of the actual position

Function

When restoring the actual position to the last buffered position, the encoder state of the active measuring system is set to "Restored". The axis can only be traversed manually.

AUTOMATIC mode

To enable NC start for the automatic execution of programs in the AUTOMATIC mode, the measuring system of the machine axis must be re-referenced.

MDI mode and overstore

In the MDI mode and for the overstore function, machining can also be performed, without referencing the axes, with restored positions. To do this, NC start with restored positions must be enabled explicitly for a specific channel:

MD20700 $MC_REFP_NC_START_LOCK = 2

Requirement

The measuring system that is active when the control is switched on must already have been referenced once before switching off.

Parameter assignment

Release: Restoration of the actual position

The enable to restore the actual position is performed by setting the encoder state to "The last buffered axis position before switching off will be restored, no automatic referencing":

MD34210 $MA_ENC_REFP_STATE[<encoder>] = 3
Release: NC START for "MDI" and "Overstore" modes

The enable of NC START for execution of part programs or part program blocks in the "MDI" and "Overstore" modes with the state "Position restored" is performed via:

MD34110 $MA_REFP_CYCLE_NR ≠ -1 (axis sequence for channel-specific referencing)
MD20700 $MC_REFP_NC_START_LOCK = 2 (NC START lock without reference point)

NC/PLC interface signals

The restored actual position is not considered to be equivalent to an actual position after reference point approach. Therefore, the state "Position restored" and not "Referenced/synchronized" is displayed for the measuring system of the machine axis.

Actual position restored:
- DB31, ... DBX60.4/.5 = 0 (referenced/synchronized 1/2)
- DB31, ... DBX71.4/.5 = 1 (position restored, encoder 1/2)

Measuring system referenced:
- DB31, ... DBX60.4/.5 = 0 → 1 (referenced/synchronized 1/2)
- DB31, ... DBX71.4/.5 = 1 → 0 (position restored, encoder 1/2)

Note

The monitoring of the traversing range limits (software limit switches, working area limitation, etc.) is already active in the "Position restored" state.

Supplementary conditions

Spindles

If the encoder limit frequency is exceeded, a spindle is reset to the "Not referenced/synchronized" state:
- DB31, ... DBX60.4/.5 = 1 → 0 (referenced/synchronized 1/2)
- DB31, ... DBX71.4/.5 = 1 → 0 (position restored, encoder 1/2)

References

Description of Functions, Basic functions, Section "R1 Referencing" > "Referencing with incremental measurement systems" (Page 1154)
15.11 Supplementary conditions

15.11.1 Large traverse range

Linear axes with a traversing range > 4096 encoder revolutions, rotatory absolute encoder EQN 1325 and a parameterized absolute encoder range of MD34220 $MA_ENC_ABS_TURNS_MODULO = 4096:

The maximum possible travel range corresponds to that of incremental encoders.

Endlessly turning rotary axes with absolute encoders:

- Any number of integer transmission ratios are permitted.
- We recommend that you parameterize endlessly turning rotary axes with absolute encoders as modulo rotary axes (traversing range 0...360 degrees):
  MD30310 $MA_ROT_IS_MODULO = 1
  Otherwise, the rotary axis may require a very large traversing path to reach absolute zero when the measuring system is activated.

Machine axes with absolute encoders:

In order that the controller correctly determines the current actual position after the restart of the measuring system, the machine axis may only be moved less than half the absolute encoder range when the measuring system is switched off:

MD34220 $MA_ENC_ABS_TURNS_MODULO

Notes on uniqueness of encoder positions

Note

Linear absolute encoders

The absolute value of linear position encoders, e.g. Heidenhain LC181, is always unique for the scale lengths available.

Rotary absolute encoders

The absolute value of rotary absolute encoders is only unique within the range of the specific maximum encoder revolutions.

For example, the EQN 1325 rotary absolute encoder by Heidenhain supplies a unique absolute value in the range of 0 to 4,096 encoder revolutions.

Depending on how the encoder is connected that will result in:

- Rotary axis with encoder on load: 4096 load revolutions
- Rotary axis with encoder on motor: 4096 motor revolutions
- Linear axis with encoder on motor: 4096 motor revolutions

Example:

An EQN 1325 rotary absolute encoder is mounted on the motor of a linear axis. For an effective leadscrew pitch of 10 mm this will result in a unique absolute value within the travel range -20.48 to +20.48 m.
15.12 Data lists

15.12.1 Machine data

15.12.1.1 NC-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11300</td>
<td>JOG_INC_MODE_LEVELTRIGGERD</td>
<td>INC/REF in jog/continuous mode</td>
</tr>
</tbody>
</table>

15.12.1.2 Channel-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20700</td>
<td>REFP_NC_START_LOCK</td>
<td>NC start disable without reference point</td>
</tr>
</tbody>
</table>

15.12.1.3 Axis/spindle-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>30200</td>
<td>NUM_ENCS</td>
<td>Number of encoders</td>
</tr>
<tr>
<td>30240</td>
<td>ENC_TYP</td>
<td>Actual value encoder type</td>
</tr>
<tr>
<td>30242</td>
<td>ENC_IS_INDEPENDENT</td>
<td>Encoder is independent</td>
</tr>
<tr>
<td>30250</td>
<td>ACT_POS_ABS</td>
<td>Absolute encoder position at time of deactivation.</td>
</tr>
<tr>
<td>30270</td>
<td>ENC_ABS_BUFFERING</td>
<td>Absolute encoder: Traversing range extension</td>
</tr>
<tr>
<td>30300</td>
<td>IS_ROT_AX</td>
<td>Rotary axis/spindle</td>
</tr>
<tr>
<td>30310</td>
<td>ROT_IS_MODULO</td>
<td>Modulo conversion for rotary axis / spindle</td>
</tr>
<tr>
<td>30330</td>
<td>MODULO_RANGE</td>
<td>Magnitude of the modulo range</td>
</tr>
<tr>
<td>30340</td>
<td>MODULO_RANGE_START</td>
<td>Starting position of modulo range</td>
</tr>
<tr>
<td>30355</td>
<td>MISC_FUNCTION_MASK</td>
<td>Axis functions</td>
</tr>
<tr>
<td>31122</td>
<td>BERO_DELAY_TIME_PLUS</td>
<td>BERO delay time in plus direction</td>
</tr>
<tr>
<td>31123</td>
<td>BERO_DELAY_TIME_MINUS</td>
<td>BERO delay time in minus direction</td>
</tr>
<tr>
<td>34000</td>
<td>REFP_CAM_IS_ACTIVE</td>
<td>Axis with reference cam</td>
</tr>
<tr>
<td>34010</td>
<td>REFP_CAM_DIR_IS_MINUS</td>
<td>Reference point approach in minus direction</td>
</tr>
<tr>
<td>34020</td>
<td>REFP_VELO_SEARCH_CAM</td>
<td>Homing approach velocity</td>
</tr>
<tr>
<td>34030</td>
<td>REFP_MAX_CAM_DIST</td>
<td>Maximum distance to reference cam</td>
</tr>
<tr>
<td>34040</td>
<td>REFP_VELO_SEARCH_MARKER</td>
<td>Reference point creep velocity</td>
</tr>
<tr>
<td>34050</td>
<td>REFP_SEARCH_MARKER_REVERSE</td>
<td>Direction reversal to reference cam</td>
</tr>
<tr>
<td>34060</td>
<td>REFP_MAX_MARKER_DIST</td>
<td>Maximum distance to reference mark; Max. distance to 2 reference mark for distance-coded scales</td>
</tr>
<tr>
<td>34070</td>
<td>REFP_VELO_POS</td>
<td>Reference point positioning velocity</td>
</tr>
</tbody>
</table>
### 15.12 Data lists

| Number | Identifier: $MA_\_
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>34080</td>
<td>REFP_MOVE_DIST</td>
</tr>
<tr>
<td>34090</td>
<td>REFP_MOVE_DIST_CORR</td>
</tr>
<tr>
<td>34092</td>
<td>REFP_CAM_SHIFT</td>
</tr>
<tr>
<td>34093</td>
<td>REFP_CAM_MARKER_DIST</td>
</tr>
<tr>
<td>34100</td>
<td>REFP_SET_POS</td>
</tr>
<tr>
<td>34102</td>
<td>REFP_SYNC_ENCS</td>
</tr>
<tr>
<td>34104</td>
<td>REFP_PERMITTED_IN_FOLLOWUP</td>
</tr>
<tr>
<td>34110</td>
<td>REFP_CYCLE_NR</td>
</tr>
<tr>
<td>34120</td>
<td>REFP_BERO_LOW_ACTIVE</td>
</tr>
<tr>
<td>34200</td>
<td>ENC_REFP_MODE</td>
</tr>
<tr>
<td>34210</td>
<td>ENC_REFP_STATE</td>
</tr>
<tr>
<td>34220</td>
<td>ENC_ABS_TURNS_MODULO</td>
</tr>
<tr>
<td>34230</td>
<td>ENC_SERIAL_NUMBER</td>
</tr>
<tr>
<td>34232</td>
<td>EVERY_ENC_SERIAL_NUMBER</td>
</tr>
<tr>
<td>34300</td>
<td>ENC_REFP_MARKER_DIST</td>
</tr>
<tr>
<td>34310</td>
<td>ENC_MARKER_INC</td>
</tr>
<tr>
<td>34320</td>
<td>ENC_INVERS</td>
</tr>
<tr>
<td>34330</td>
<td>REFP_STOP_AT_ABS_MARKER</td>
</tr>
<tr>
<td>35150</td>
<td>SPIND_DES_VELO_TOL</td>
</tr>
<tr>
<td>36302</td>
<td>ENC_FREQ_LIMIT_LOW</td>
</tr>
<tr>
<td>36310</td>
<td>ENC_ZERO_MONITORING</td>
</tr>
</tbody>
</table>

### 15.12.2 Signals

#### 15.12.2.1 Signals to BAG

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode group RESET</td>
<td>DB11.DBX0.7</td>
<td>DB3000.DBX0.7</td>
</tr>
<tr>
<td>Machine function REF</td>
<td>DB11.DBX1.2</td>
<td>DB3000.DBX1.2</td>
</tr>
</tbody>
</table>

#### 15.12.2.2 Signals from BAG

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active machine function REF</td>
<td>DB11.DBX5.2</td>
<td>DB3100.DBX1.2</td>
</tr>
</tbody>
</table>
### 15.12.2.3 Signals to channel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate referencing</td>
<td>DB21, ... DBX1.0</td>
<td>DB320x.DBX1.0</td>
</tr>
<tr>
<td>OEM channel signal (HMI → PLC) REF</td>
<td>DB21, ... DBX28.7</td>
<td>-</td>
</tr>
</tbody>
</table>

### 15.12.2.4 Signals from channel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referencing active</td>
<td>DB21, ... DBX33.0</td>
<td>DB330x.DBX1.0</td>
</tr>
<tr>
<td>Reset</td>
<td>DB21, ... DBX35.7</td>
<td>DB330x.DBX3.7</td>
</tr>
<tr>
<td>All axes referenced</td>
<td>DB21, ... DBX36.2</td>
<td>DB330x.DBX4.2</td>
</tr>
</tbody>
</table>

### 15.12.2.5 Signals to axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow-up mode (request)</td>
<td>DB31, ... DBX1.4</td>
<td>DB380x.DBX1.4</td>
</tr>
<tr>
<td>Position measuring system 1 / position measuring system 2</td>
<td>DB31, ... DBX1.5-6</td>
<td>DB380x.DBX1.5-6</td>
</tr>
<tr>
<td>Reference point value 1 to 4</td>
<td>DB31, ... DBX2.4-7</td>
<td>DB380x.DBX2.4-7</td>
</tr>
<tr>
<td>Traversing keys minus/plus</td>
<td>DB31, ... DBX4.6-7</td>
<td>DB380x.DBX4.6-7</td>
</tr>
<tr>
<td>Deceleration of reference point approach</td>
<td>DB31, ... DBX12.7</td>
<td>DB380x.DBX1000.7</td>
</tr>
</tbody>
</table>

### 15.12.2.6 Signals from axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referenced, synchronized 1/2</td>
<td>DB31, ... DBX60.4-5</td>
<td>DB390x.DBX0.4-5</td>
</tr>
<tr>
<td>Follow up active</td>
<td>DB31, ... DBX61.3</td>
<td>DB390x.DBX1.3</td>
</tr>
<tr>
<td>Traversing command minus/plus</td>
<td>DB31, ... DBX64.6-7</td>
<td>DB390x.DBX4.6-7</td>
</tr>
<tr>
<td>Restored 1/2</td>
<td>DB31, ... DBX71.4-5</td>
<td>DB390x.DBX11.4-5</td>
</tr>
</tbody>
</table>
R1: Referencing

15.12 Data lists
16.1 Brief Description

The primary function of a spindle is to set a tool or workpiece in rotary motion in order to facilitate machining.

Depending on the type of machine, the spindle must support the following functions in order to achieve this:

- Input of a direction of rotation for the spindle (M3, M4)
- Input of a spindle speed (S, SVC)
- Spindle stop, without orientation (M5)
- Spindle stop with orientation / Spindle positioning (SPOS, M19 and SPOSA)
- Gear change (M40 to M45)
- Spindle axis functionality (spindle becomes rotary axis and vice versa)
- Thread cutting (G33, G34, G35)
- Tapping without compensating chuck (G331, G332)
- Tapping with compensating chuck (G63)
- Revolutinal feedrate (G95)
- Constant cutting rate (G96, G961, G97, G971)
- Programmable spindle speed limits (G25, G26, LIMS=)
- Position encoder assembly on the spindle or on the spindle motor
- Spindle monitoring for min. and max. speed as well as max. encoder limit frequency and end point monitoring of spindle
- Switching the position control (SPCON, SPCOF, M70) on/off
- Programming of spindle functions:
  - From the part program
  - Via synchronized actions
  - Via PLC with FC18 or via special spindle interfaces for simple spindle activation
16.2 Modes

16.2.1 Overview

Spindle modes

The spindle can have the following modes:

- Control mode
- Oscillation mode
- Positioning mode
- Synchronous mode, synchronous spindle
  References: Function Manual, Extension Functions, Synchronous Spindle (S3)
- Rigid tapping
  References: Programming Manual, Fundamentals; Chapter: Motion commands

Axis mode

The spindle can be switched from spindle mode to axis mode (rotary axis) if the same motor is used for spindle and axis operation.
16.2.2 Mode change

Switching between spindle and axis operation can be done as follows:

- **Control mode → Oscillation mode**
  The spindle changes to oscillation mode if a new gear stage has been specified using automatic gear step selection (M40) in conjunction with a new S value or by M41 to M45. The spindle only changes to oscillation mode if the new gear step is not equal to the current actual gear step.

- **Oscillation mode → Control mode**
  When the new gear is engaged, the interface signal: DB31, ... DBX84.6 (Oscillation mode)
  is reset and the interface signal: DB31, ... DBX16.3 (Gear changed)
  is used to go to control mode.
  The last programmed spindle speed (S value) is reactivated.

- **Control mode → Positioning mode**
  To stop the spindle from rotation (M3 or M4) with orientation or to reorient it from standstill (M5), SPOS, M19 or SPOSA are used to switch to positioning mode.

- **Positioning mode → Control mode**
  M3, M4 or M5 are used to change to control mode if the orientation of the spindle is to be terminated. The last programmed spindle speed (S value) is reactivated.

- **Positioning mode → Oscillation mode**
  If the orientation of the spindle is to be terminated, M41 to M45 can be used to change to oscillation mode. When the gear change is complete, the last programmed spindle speed (S value) and M5 (control mode) are reactivated.
• Positioning mode → Axis mode
  If a spindle was stopped with orientation, the assigned axis name is used to program a
t change to axis mode. The gear step is retained.

• Control mode → Axis mode
  Switching from control mode to axis mode can be also achieved by the programming of
  M70. In this case, a rotating spindle is decelerated in the same way as for M5, position
  control activated and the zero parameter set selected.

• Axis mode → Control mode
  To terminate axis mode, M3, M4 or M5 can be used to change to control mode. The last
  programmed spindle speed (S value) is reactivated.

• Axis mode → Oscillation mode
  To terminate axis mode, M41 to M45 can be used to change to oscillation mode (only if the
  programmed gear step is not the same as the actual gear step). When the gear change is
  complete, the last programmed spindle speed (S value) and M5 (control mode) are
  reactivated.

16.2.3 Control mode

When open-loop control mode?

The spindle is in open-loop control mode with the following functions:

• Constant spindle speed:
  – S... M3/M4/M5 and G93, G94, G95, G97, G971
  – S... M3/M4/M5 and G33, G34, G35
  – S... M3/M4/M5 and G63

• Constant cutting speed:
  – G96/G961 S... M3/M4/M5

The spindle need not be synchronized.

Requirements

A spindle position actual value encoder is absolutely essential for M3/M4/M5 in connection
with:

• Revolitional feedrate (G95)
• Constant cutting speed (G96, G961, G97, G971)
• Thread cutting (G33, G34, G35)
• Tapping without compensating chuck (G331, G332)
• Activate position control (SPCON, M70)
A spindle position actual value encoder is not required for M3/M4/M5 in connection with:

- Inverse-time feedrate coding (G93)
- Feedrate in mm/min or inch/min (G94)
- Tapping with compensating chuck (G63)

**Speed control mode**

Speed control mode is particularly suitable if a constant spindle speed is required, but the position of the spindle is not important (e.g. constant milling speed for even appearance of the workpiece surface).

- Speed control mode is activated in the part program with M3, M4, M5 or with SPCOF.
- The following NC/PLC interface signal is set: DB31, ... DBX84.7 (control mode)
- NC/PLC IS: DB31, ... DBX61.5 (position controller active) is reset if position control is not used.
- Acceleration in speed control mode is defined independently of the gear stage in the machine data:
  - MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL
  - The value should reflect the physical circumstances, if possible.

**Position control mode**

Position control is particularly suitable if the position of the spindle needs to be tracked over a longer period or if synchronous spindle setpoint value linkage is to be activated.

- Position control mode is switched on in the part program with: SPCON(<spindle number>)
- The following NC/PLC interface signal is set: DB31, ... DBX61.5 (position controller active)
- Acceleration in position control mode is defined independent of the gear stage in the machine data:
  - MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

**Independent spindle reset**

The spindle response after a reset or the end of the program (M2, M30) is set with the machine data:
MD35040 $MA_SPIND_ACTIVE_AFTER_RESET (individual spindle reset)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>When the spindle is reset or at the end of the program, the spindle immediately decelerates to a stop at the active acceleration rate. The last programmed spindle speed and direction of rotation are deleted.</td>
</tr>
<tr>
<td>1</td>
<td>Upon reset or at the end of the program, the last programmed spindle speed (S-value) and the last programmed direction of spindle rotation (M3, M4, M5) are retained. The spindle is not braked.</td>
</tr>
</tbody>
</table>

If prior to reset or end of program the constant cutting speed (G96, G961) is active, the current spindle speed (in relation to 100% spindle override) is internally accepted as the spindle speed last programmed.

The spindle can only be stopped with the NC/PLC interface signal:

DB31, ... DBX2.2 (delete distance-to-go / spindle reset)

The direction of rotation is deleted in the event of all alarms generating a spindle quick stop. The last programmed spindle speed (S value) is retained. Once the source of the alarm has been eliminated, the spindle must be restarted.

**Spindle actual speed display and spindle behavior with G96, G961**

**DB31, ... DBX61.4 (axis/spindle stationary)**

The speed at which the spindle is deemed to be "stationary" is set with the machine data:

MD36060 $MA_STANDSTILL_VELO_TOL

The value should be measured in such a way that the following NC/PLC interface signal is reliably present at a standstill:

DB31, ... DBX61.4 (axis/spindle stationary)

If DB31, ... DBX61.4 (axis/spindle stationary) is signaled and there is no closed-loop position control active for the spindle, an actual speed of zero is displayed at the user interface, and zero is read with the system variable $AA_S[<n>].

**Spindle response at constant cutting speed G96, G961**

- At the start of machining (transition from G0 to Gx) and after NC stop, G60 (exact stop, modal) and G9 (exact stop, non-modal) the system waits until the actual speed has reached the speed setpoint tolerance range before starting the path.

  DB31, ... DBX83.5 (nact = nset)

- The NC/PLC IS:

  DB31, ... DBX83.5 (nact = nset)

  and

  DB31, ... DBX83.1 (setpoint speed limited) are also set to defined values even if significant speed changes are specified (transverse axis travels towards position 0).
● If the speed drops below the minimum speed or when NC/PLC IS:
   DB31, ... DBX61.4 (axis/spindle stationary) is detected, NC/PLC IS:
   DB31, ... DBX83.5 (nact = nset) is reset (e.g. for an emergency machine strategy).

● A path operation which has started (G64, rounding), is not interrupted.

In addition, the spindle response is influenced by the following machine data:

MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START (feed enable with spindle in setpoint range).

**Spindle behavior at the end of gear stage change**

● NC/PLC IS:
   DB31, ... DBX16.3 (gear changed)
   informs the NC that the new gear stage (NC/PLC IS DB31, ... DBX16.0-16.2 (actual gear stage A to C)) applies and oscillation mode is terminated.
   In this case, it does not matter whether NC/PLC IS:
   DB31, ... DBX18.5 (oscillation mode) is still set.
   The actual gear stage should correspond to the set gear stage.
   The actual gear stage signaled is relevant for selection of the parameter set.

● Once the gear stage change (GSW) has been acknowledged via the PLC (DB31, ... DBX16.3), the spindle is in speed control mode (DB31, ... DBX84.7 = 1).
   If a direction of rotation (M3, M4, M5 or FC18: "Start spindle rotation") or a spindle speed (S value) was programmed before the gear stage change, then the last speed and direction of rotation will be reactivated after the gear stage change.

16.2.4 Oscillation mode

Oscillation mode is activated for the spindle during the gear step change.

The mode of operation is described in detail in the topic "Gear step change with oscillation mode (Page 1251)".

16.2.5 Positioning mode

16.2.5.1 General functionality

**When is positioning mode used?**

The spindle positioning mode stops the spindle at the defined position and activates the position control, which remains active until it is de-activated.
For the following functions the spindle is in positioning mode:

- \text{SPOS[<n>]=}...
- \text{SPOS[<n>]=}ACP(...)
- \text{SPOS[<n>]=}ACN(...)
- \text{SPOS[<n>]=}AC(...)
- \text{SPOS[<n>]=}IC(...)
- \text{SPOS[<n>]=}DC(...) equal to \text{SPOSA[<n>]=}...
- \text{M19 or M[<n>]=}19

The address extension \([<n>]\) with \(<n> = \) spindle number may not apply for the main spindle.

\text{SPOS[<n>]=}AC(...)

Spindle positioning to an absolute position (0 to 359.999 degrees). The positioning direction is determined either by the current direction of spindle rotation (spindle rotating) or the distance-to-go.

\text{SPOS[<n>]=}IC(...)

Spindle positioning to an incremental position (+/- 999999.99 degrees) in relation to the last programmed position. The positioning direction is defined by the sign of the path to be traversed.

\text{SPOS[<n>]=}DC(...)

Spindle positioning across the shortest path to an absolute position (0 to 359.999 degrees). The positioning direction is determined either by the current direction of spindle rotation (spindle rotating) or automatically by the control (spindle stationary).

\text{SPOS[<n>]=}...

Same functional sequence as \text{SPOS[<n>]=}DC(...).

\text{SPOS[<n>]=}ACP(...)

Approaches the position from the positive direction.

When positioning from a negative direction of rotation, the speed is decelerated to zero and accelerated in the opposite direction to execute the positive approach.
SPOS[<n>]=ACN(...)  
Approaches the position from the negative direction.  
When positioning from a positive direction of rotation, the speed is decelerated to zero and accelerated in the opposite direction to execute the negative approach.

M19 (DIN 66025)  
M19 can be used to position the spindle. The position and the position approach mode are read here from the following setting data:
SD43240 $SA_M19_SPOS[<n>] (spindle position for spindle positioning with M19)  
SD43250 $SA_M19_SPOSMODE[<n>] (spindle position for spindle positioning with M19)  
The positioning options of M19 are identical to those of:
SPOS = <approach mode> <position/path>  
M19 is output as an auxiliary function to the NC/PLC interface as an alternative to M3, M4, M5, and M70. The M19 block remains active in the interpolator for the duration of positioning (like SPOS).  
Part programs using M19 as a macro (e.g. DEFINE M19 AS SPOS = 0) or as a subprogram, continue to remain executable. For the sake of compatibility with previous controls, the internal processing of M19 (NCK positions the spindle) can be disabled as shown in the following example:
MD22000 $MC_AUXFU_ASSIGN_GROUP[0] = 4 ; Auxiliary function group: 4  
MD22010 $MC_AUXFU_ASSIGN_TYPE[0] = "M" ; Auxiliary function type: "M"  
MD22020 $MC_AUXFU_ASSIGN_EXTENSION[0] = 0 ; Auxiliary functions Extension: 0  
MD22030 $MC_AUXFU_ASSIGN_VALUE[0] = 19 ; Auxiliary function value: 19

Implicitly generated auxiliary function M19  
To achieve uniformity in terms of how M19 and SPOS or SPOSA behave at the NC/PLC interface, auxiliary function M19 can be output to the NC/PLC interface in the event of SPOS and SPOSA.  
The two following options are available for activating this function:  
- Channel-specific activation for all the spindles in the channel via the machine data:  
  MD20850 $MC_SPOS_TO_VDI (Output of M19 to the PLC with SPOS/SPOSA)  

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>If bit 19 is also set to &quot;0&quot; in the MD35035 $MA_SPIND_FUNCTION_MASK, no auxiliary function M19 is generated in SPOS and SPOSA. This therefore eliminates the acknowledgement time for the auxiliary function.</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>The auxiliary function M19 is generated and output to the PLC during the programming of SPOS and SPOSA in the part program. The address extension corresponds to the spindle number.</td>
</tr>
</tbody>
</table>
Spindle-specific and cross-channel activation via the machine data:
MD35035 $MA_SPIND_FUNCTION_MASK (spindle functions)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>0</td>
<td>If bit 0 is also set to &quot;0&quot; in the MD20850 $MC_SPOS_TO_VDI, no auxiliary function M19 is generated in SPOS and SPOSA. This therefore eliminates the acknowledgement time for the auxiliary function.</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>The implicit auxiliary function M19 is generated and output to the PLC during the programming of SPOS and SPOSA. The address extension corresponds to the spindle number.</td>
</tr>
</tbody>
</table>

**Note**
Activation via MD35035 should be preferred when using a spindle in multiple channels (axis/spindle exchange).

The auxiliary function M19 is implicitly generated if either of the MD configurations = 1.
After activation, the minimum duration of an SPOS/SPOSA block is increased to the time for output and acknowledgement of the auxiliary functions by the PLC.
The properties of the implicitly generated auxiliary function output M19 are "Quick" and "Output during motion". These properties are fixed settings and are independent of the M19 configuration in the auxiliary function-specific machine data (MD..._$M..._AUXFU...).
There is no auxiliary function M19 implicitly generated in the case of spindle positioning commands via FC 18.

**End of positioning**
The positioning can be programmed with:

- **FINEA[S<n>]:** End of motion on reaching "Exact stop fine" (DB31, ... DBX60.7)
- **COARSEA[S<n>]:** End of motion on reaching "Exact stop coarse" (DB31, ... DBX60.6)
- **IPOENDA[S<n>]:** End of motion on reaching "IPO stop"

In addition, an end-of-motion criterion for block changes can be set in the braking ramp (100-0%) with IPOBRKA for single-axis interpolation.

**References:**
Function Manual, Extended Functions; Positioning Axes (P2)

**Block change**
The program advances to the next block if the end-of-motion criteria for all spindles or axes programmed in the current block, plus the block change criterion for path interpolation, are fulfilled. This applies to both part-program and technology-cycle blocks.
SPOS, M19 and SPOSA have the same functionality but differ in their block change behavior:

- **SPOS and M19**
  The block change is carried out if all functions programmed in the block have reached their end-of-block criterion (e.g. all auxiliary functions acknowledged by the PLC, all axes have reached their end points) and the spindle has completed its positioning motion.

- **SPOSA**
  The program moves to the next block if all the functions (except for spindle) programmed in the current block have reached their end-of-block criterion. If SPOSA is the only entry in the block, block change is performed immediately. The spindle positioning operation may be programmed over several blocks (see WAITS).

### Coordination

A coordination of the sequence of motions can be achieved with:

- PLC
- MD configuration
- Programming in the part program

#### PLC

If the NC/PLC interface signal:
DB31, ... DBX83.5 (spindle in the setpoint range)
is not available, then the channel-specific NC/PLC interface signal:
DB21, ... DBX 6.1 (read-in inhibit)
can be set in order to wait for a spindle to reach a certain position.

#### MD configuration

Setting:
MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START = 1
is used to perform path interpolation taking the tolerance:
MD35150 $MA_SPIND_DES_VELO_TOL
into account only if the spindle has rotated up to the preselected speed.

The setting
MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START = 2
is used to stop traveling path axes before the start of machining at the last G0.

Machining continues:

- With the next traversing command.
- If the spindle speed is reached.
- When MD35510 $MA_SPIND_STOPPED_AT_IPO_START = 1
  (path feed enable, if spindle stationary).

#### Programming in the part program
Coordination actions in the part program have the following advantages:

- The part program author can decide at what point in the program the spindle needs to be up to speed, e.g. in order to start machining a workpiece.
- This avoids unnecessary delays.

Coordination in the part program involves programming the `WAITS` command:

- `WAITS:` for main spindle (master spindle)
- `WAITS[n]:` for spindles with number `<n>`
- `WAITS[n,m,...]:` for several spindles up to the maximum number of spindles

<table>
<thead>
<tr>
<th>CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordination error</td>
</tr>
</tbody>
</table>
| The part program author must ensure that one of the following maintenance conditions occurs for `WAITS`.
- Position reached
- Spindle stationary
- Spindle up to programmed speed

In cases where one spindle is used in several channels, the part program author must ensure that `WAITS` starts at the earliest in the phase in which the spindle from another channel has already started to accelerate or decelerate towards the required new speed or direction of rotation.

The control waits before executing subsequent blocks until:

- Position(s) programmed with `SPOSA` are reached.
- Spindle standstill is reached with M5:
  - `DB31, ..., DBX 61.4` (spindle stationary)
  - taking into account the tolerance:
  - `MD36060 $MA_STANDSTILL_VELO_TOL`
  - `WAITS` is terminated and the next block loaded when the first occurrence of the signal is detected.
- In M3/M4 (speed control mode), the speed in the setpoint range is:
  - `DB31, ..., DBX83.5` (spindle in setpoint range)
  - taking into account the tolerance:
  - `MD35150 $MA_SPIND_DES_VELO_TOL`
  - `WAITS` is terminated and the next block loaded when the first occurrence of the signal is detected.

This `WAITS` function applies in the programmed channel.

`WAITS` can be used to wait for all spindles known to this channel, although spindles may also have been started in other channels.
Special cases

- **Tolerance for spindle speed:**
  If the machine data setting is:
  
  MD35150 \$MA\_SPIND\_DES\_VELO\_TOL = 0
  
  the NC/PLC interface signal:
  
  DB31, ... DBX83.5 (spindle in setpoint range)
  
  is always set to 1.

  WAITS is terminated as soon as the spindle has reached the setpoint-side target after a
  change in speed or direction (M3/M4).

- **Missing enable signals:**
  If the WAITS function waits for the "Spindle in setpoint range" signal in speed control mode
  and the spindle stops or fails to rotate because an enable signal (axial feed enable,
  controller, pulse enable, etc.) is missing, the block is not terminated until the "Spindle in
  setpoint range" signal is active, once enable signals are being received again.

- **Response to NC and mode-group stop:**
  If an NC or mode-group stop is triggered during WAITS, the wait operation is resumed after
  the NC start with all the above conditions.

**Note**

In particular when using spindles across different channels, care should be taken when
programming not to start WAITS too early in one channel, i.e. at a time when the spindle
in the other channel is still rotating at its "old" speed.

In such cases, the "Spindle in setpoint range" signal is activated and WAITS is stopped too
soon.

To prevent this happening, it is strongly recommended to set a WAITM before WAITS.

Feedrate

The positioning speed is configured in the machine data:

 MD35300 \$MA\_SPIND\_POSCTRL\_VELO (position control switching speed)

The configured positioning speed can be modified by programming or by synchronized actions:

 FA[S\(<n>\)]=<value>

where: \(<n>\): Spindle number

<value>: Positioning speed in degrees/min

With \(FA[S\(<n>\)]=0\), the configured speed takes effect.

Acceleration

The accelerations are configured in the machine data:

 MD35210 \$MA\_GEAR\_STEP\_POSCTRL\_ACCEL (acceleration in position control mode)

 MD35200 \$MA\_GEAR\_STEP\_SPEEDCTRL\_ACCEL (acceleration in the speed control mode)

The configured dynamic response during positioning can be modified by programming or by
synchronized actions:

 ACC[S\(<n>\)]=<value>
where: \(<n>\): Spindle number

\(<\text{value}>\): Acceleration as a percentage of the configured acceleration

With \(\text{ACC}[S<n>]=0\), the configured acceleration takes effect.

**Aborting the positioning process**

The positioning action is aborted:

- By the NC/PLC interface signal:
  DB31, ... DBX2.2 (delete distance-to-go / spindle reset).
- With every reset (e.g. operator panel front reset).
- Through NC stop.

The abort response is independent of the machine data:

MD35040 $MA\_SPIND\_ACTIVE\_AFTER\_RESET$ (individual spindle reset)

**Special features**

The spindle override switch is active.

**16.2.5.2 Positioning from rotation**

**Initial situation**

The spindle can be in speed control mode or in position control mode when positioning starts (\(SPOS, M19\) or \(SPOSA\) command in the program).

One must distinguish between the following cases:

<table>
<thead>
<tr>
<th>Case 1: Spindle in speed control mode, encoder limit frequency exceeded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 2: Spindle in speed control mode, encoder limit frequency not exceeded</td>
</tr>
<tr>
<td>Case 3: Spindle in position control mode</td>
</tr>
<tr>
<td>Case 4: Spindle speed ≤ Position-control activation speed</td>
</tr>
</tbody>
</table>
Procedure

Figure 16-1  Positioning from rotation

**Note**

The speed arising from the configuration of the encoder limit frequency for the resynchronization of the encoder (MD36302 $MA_ENC_FREQ_LOW) must be greater than the position-control activation speed (MD35300 $MA_SPIND_POSCTRL_VELO).

**Phase 1**

**Positioning from phase 1a:**

The spindle is rotating at a higher speed than the encoder limit frequency. The spindle is not synchronized.

**Positioning from phase 1b:**

The spindle is rotating at a lower speed than the encoder limit frequency. The spindle is synchronized.

**Note**

If the position control is active, the speed can only amount to 90% of the maximum speed of the spindle or the encoder limit frequency (10% control reserve required).

**Positioning from phase 1c:**
The spindle rotates at the programmed spindle speed whereby the speed is lower than the configured position-control activation speed:

\[
\text{MD35300 } \text{\$MA\_SPIND\_POSCTRL\_VELO}
\]

The spindle is synchronized.

**Phase 2**

**Spindle speed > Position-control activation speed**

When the SPOS, M19 or SPOSA command is activated, the spindle begins to slow down to the position-control activation speed with the configured acceleration:

\[
\text{MD35200 } \text{\$MA\_GEAR\_STEP\_SPEEDCTL\_ACCEL}
\]

The spindle is synchronized once the encoder limit frequency threshold is crossed.

**Spindle speed < Position-control activation speed**

SPOS, M19 or SPOSA are programmed to switch the spindle to position control mode (if it is not already in that mode).

The configured acceleration in position control mode is activated:

\[
\text{MD35210 } \text{\$MA\_GEAR\_STEP\_POSCTRL\_ACCEL}
\]

The travel path to the target point is calculated.

The spindle travels to the programmed end point optimally in terms of time. This means that the end point is approached at the highest possible speed (maximum MD35300 $MA\_SPIND\_POSCTRL\_VELO). Depending on the appropriate secondary conditions, phases 2 - 3 - 4 - 5 or 4a - 5a are executed.

**Phase 3**

**Spindle speed > Position-control activation speed**

When the configured position-control activation speed (MD35300 $MA\_SPIND\_POSCTRL\_VELO) is reached:

- Position control is activated (if not already active).
- The distance to go (to the target point) is calculated.
- There is a switch to the configured configured acceleration in position control mode (or this acceleration is retained):

\[
\text{MD35210 } \text{\$MA\_GEAR\_STEP\_POSCTRL\_ACCEL}
\]

**Spindle speed < Position-control activation speed**

The spindle was accelerated up to the configured position-control activation speed (MD35300 $MA\_SPIND\_POSCTRL\_VELO) to reach the end point. This is not exceeded.

The braking start point calculation detects when the programmed spindle position can be approached accurately at the acceleration configured in position control mode (MD35210 $MA\_GEAR\_STEP\_POSCTRL\_ACCEL).
Phase 4

**Spindle speed > Position-control activation speed**

The spindle brakes from the calculated "braking point" with machine data: MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL to the target position.

**Spindle speed < Position-control activation speed**

At the time identified by the braking start point calculation in phase 3, the spindle brakes to a standstill with the acceleration configured in position control mode (MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL).

**Phase 4a:**

When the SPOS command is activated the proximity of the end point is such that the spindle can no longer be accelerated to the configured position-control activation speed (MD35300 $MA_SPIND_POSCTRL_VEL0).

The spindle brakes to a standstill with the acceleration configured in position control mode (MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL).

Phase 5

**Spindle speed > Position-control activation speed**

Position control remains active and holds the spindle in the programmed position.

**Note**

The maximum encoder limit frequency of the spindle position actual-value encoder is monitored by the control (it may be exceeded); in position control mode, the setpoint speed is reduced to 90% of the measuring system limit speed.

The following NC/PLC interface signal is set:

DB31, ... DBX83.1 (programmed speed too high)

If "MS limit frequency exceeded" is still pending following a reduction in the setpoint speed, an alarm is output.

**Spindle speed < Position-control activation speed (Phase 5, 5a)**

The spindle is stationary and it has reached the position. The position control is active and stops the spindle in the programmed position.

If the distance between the spindle actual position and the programmed position (spindle setpoint position) is less than the configured exact stop fine and coarse limits, the following NC/PLC interface signals are set:

DB31, ... DBX60.7 (Position reached with coarse exact stop)
DB31, ... DBX60.7 (Position reached with fine exact stop)

The exact stop limits are defined with the machine data:

MD36010 $MA_STOP_LIMIT_FINE (exact stop fine)
MD36000 $MA_STOP_LIMIT_COARSE (exact stop coarse)

**Note**

The positioning procedure is considered complete when the end-of-positioning criterion is reached and signaled.

The condition is "Exact stop fine". This applies to SPOS, M19 or SPOSA from the part program, synchronized actions and spindle positioning by the PLC using FC 18.
16.2.5.3 Positioning from standstill

Procedure

A distinction is made between two cases with regard to positioning from standstill:

- **Case 1:** The spindle is not synchronized.
  This is the case if the spindle is to be positioned after switching on the control and drive or after a gear step change (e.g. for a tool change).
  MD31040 $MA_ENC_IS_DIRECT = 0

- **Case 2:** The spindle is synchronized.
  This is the case if, after switching on the control and drive, the spindle is to be rotated through a minimum of one revolution with M3 or M4 and then stopped with M5 (synchronization with the zero mark) before the first positioning action.

Figure 16-2 Positioning with stationary spindle
Phase 1

**Case 1: Spindle not synchronized**

With the programming of SPOS, M19 or SPOSA the spindle accelerates with the acceleration from the machine data:

MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL (Acceleration in the speed control mode)

This direction of rotation is defined by the machine data:

MD35350 $MA_SPIND_POSITIONING_DIR (Direction of rotation while positioning to standstill)

Exception:

If ACN, ACP, IC is used for positioning, the programmed direction of travel is activated.

The spindle is synchronized at the next zero mark of the spindle position actual-value encoder and switches to the position control mode.

Whether the zero mark is found in the traversed path (except for IC), is monitored:

MD34060 $MA_REFP_MAX_MARKER_DIST (maximum distance to the reference mark)

If the speed defined in machine data:

MD35300 $MA_SPIND_POSCTRL_VELO (Positioning speed) is reached before the spindle is synchronized, the spindle will continue to rotate at the positioning activation speed (it is not accelerated further).

**Case 2: Spindle synchronized**

SPOS, M19 or SPOSA will switch the spindle to position control mode.

The acceleration from the following machine data is active:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL (acceleration in position control mode)

The direction of rotation is defined by the programmed motion (ACP, ACN, IC, DC) or via the pending distance-to-go.

The speed entered in

MD35300 $MA_SPIND_POSCTRL_VELO (position control activation speed) is not exceeded in the machine data.

The travel path to the end position is calculated.

The spindle travels to the programmed end point optimally in terms of time. This means that the end point is approached at the highest possible speed (maximum MD35300 $MA_SPIND_POSCTRL_VELO). Depending on the appropriate secondary conditions, the phases 1 - 2 - 3 - 4 or 1- 3a - 4a are executed.

Phase 2

**Case 1: Spindle not synchronized**

When the spindle is synchronized, position control is activated.

The spindle rotates at the maximum speed stored in machine data:

MD35300 $MA_SPIND_POSCTRL_VELO
until the braking start point calculation identifies the point at which the programmed spindle position can be approached accurately with the defined acceleration.

**Case 2: Spindle synchronized**

To reach the end point, the spindle is accelerated up to the speed defined in machine data: MD35300 $MA_SPIND_POSCTRL_VELO.

This is not exceeded.

The braking start point calculation identifies when the programmed spindle position can be approached accurately at the acceleration defined in machine data: MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL.

At the point, which is determined by the braking start point calculation in Phase 1, the spindle decelerates to a standstill with the acceleration given in the following machine data:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

**Phase 3**

At the point, which is determined by the braking start point calculation in Phase 2, the spindle decelerates to a standstill with the acceleration given in the following machine data:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

**Phase 3a:**

When the SPOS command is activated the proximity of the end point is such that the spindle can no longer be accelerated up to machine data: MD35300 $MA_SPIND_POSCTRL_VELO.

The spindle is braked to a standstill with the acceleration given in the following machine data:

MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

**Phase 4, 4a**

The spindle is stationary and it has reached the position. The position control is active and stops the spindle in the programmed position.

NC/PLC IS:
DB31, ... DBX60.6 (Position reached with exact stop coarse)
and
DB31, ... DBX60.7 (Position reached with exact stop fine)
are set if the distance between the spindle actual position and the programmed position (spindle setpoint position) is less than the settings for the exact stop fine and coarse limits.

This is defined in the machine data:

MD36010 $MA_STOP_LIMIT_FINE
MD36000 $MA_STOP_LIMIT_COARSE

**Phase 3:**
At the point, which is determined by the braking start point calculation in Phase 2, the spindle decelerates to a standstill with the acceleration given in the following machine data:

MD35210 $MA_GEAR_STEP_ POSCTRL_ACCEL

**Phase 4:**

The spindle is stationary and it has reached the position. The position control is active and stops the spindle in the programmed position.

NC/PLC IS:
DB31, ... DBX60.6 (Position reached with exact stop coarse)
and
DB31, ... DBX60.7 (Position reached with exact stop fine)
are set if the distance between the spindle actual position and the programmed position (spindle setpoint position) is less than the settings for the exact stop fine and coarse limits.

This is defined in the machine data:

MD36010 $MA_STOP_LIMIT_FINE
MD36000 $MA_STOP_LIMIT_COARSE

### 16.2.5.4 "Spindle in position" signal for tool change

**Function**

The motion sequence for a tool change, especially for milling machines, mainly comprises positioning the spindle and then the subsequent (for optimization runs, also at the same time) approach to the tool change position with the path axes. In this case, the mandatory requirement is that the spindle is reached before approaching the tool change position.

If the tool change cycle is interrupted by the machine operator (e.g. with an NC stop, NC stop axes plus spindles, mode group stop, etc.), then it must be completely ruled out that the spindle moves into the tool changer at an incorrect position.

This is the reason that for spindle positioning, when the last programmed spindle position is reached with "Exact stop fine" the following NC/PLC interface signal is output to check the position:

DB31, ... DBX85.5 (spindle in position)

**Note**

The signal is only output for the "Spindle positioning" function.

This includes:
- SPOS, SPOSA and M19 in the part program
- SPOS and M19 in synchronized actions
- Spindle positioning, using FC18
- Spindle positioning via PLC interface (DB31, ... DBX30.4)
Setting the signal

Requirements for output of signal DB31, ... DBX85.5 (spindle in position) are as follows:

- The reference state of the spindle:
  DB31, ... DBX60.4/5 (referenced/synchronized 1/2) = 1

**Note**
When positioning the spindle, the zero mark is automatically searched for. This is the reason that for an error-free sequence, the referenced signal is always available at the end of positioning movement.

- "Exact stop fine" must have been reached:
  DB31, ... DBX60.7 (exact stop fine) = 1
  Additionally, the last programmed spindle position must have been reached on the setpoint side.

Deleting the signal

When signal DB31, ... DBX60.7 is withdrawn (exact stop fine), then signal DB31, ... DBX85.5 (spindle in position) is also always reset.

Additional properties

- If the spindle is already at the programmed position after a positioning, then the NC/PLC interface signal DB31, ... DBX85.5 (spindle in position) remains set.

- If, after a positioning ("Spindle in position" signal was output) the spindle is traversed, e.g. in the JOG mode, then the NC/PLC interface signal DB31, ... DBX85.5 (spindle in position) is deleted.
  If the spindle is returned to its original position in this mode, then the NC/PLC interface signal DB31, ... DBX85.5 (spindle in position) is set again. The last position selection is kept.

16.2.6 Axis mode

16.2.6.1 General functionality

Functionality

If for certain machining tasks, e.g. on lathes with end-face machining, it is not sufficient to traverse the spindle exclusively under speed control via M3, M4, M5 or to position with SPOS, M19 or SPOSA, the spindle can be switched to position-controlled axis mode and traversed as a rotary axis.

Examples of rotary axis functions:

- Programming with axis name
- Zero offsets (G54, G55, TRANS, etc.)
● G90, G91, IC, AC, DC, ACP, ACN
● Kinematic transformations (e.g. TRANSMIT)
● Path interpolation
● Traversing as positioning axis

References:
Function Manual, Extended Functions; Section "Rotary axes (R2)"

Preconditions

● The same spindle motor is used for spindle mode and axis mode.
● The same position measurement system or separate position measurement systems can be used for spindle mode and axis mode.
● An actual position value encoder is a mandatory requirement for axis mode.
● The spindle must be referenced for use of the axis mode, e.g. referenced with G74.

Example:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>M70</td>
<td>; Switch spindle over to axis mode</td>
</tr>
<tr>
<td>G74 C1=0 Z100</td>
<td>; Reference axis</td>
</tr>
<tr>
<td>G0 C180 X50</td>
<td>; Traverse axis position-controlled</td>
</tr>
</tbody>
</table>

Configurable M function

The M function used to switch the spindle to axis mode can be configured channel-specifically via the following machine data:

MD20094 $MC_SPIND_RIGID_TAPPING_M_NR

Note

The controller detects the transition to axis mode automatically from the program sequence (see "Implicit transition to axis mode (Page 1224)"). The explicit programming of the configured M function for switching the spindle to axis mode in the part program is therefore not necessary. However, the M function can continue to be programmed, e.g. to increase the readability of the part program.

Special features

● The feed override switch is active.
● The NC/PLC interface signal does not terminate the axis mode per default: DB21, ... DBX7.7 (reset).
● The NC/PLC interface signals:
  DB31, ... DBB16 to DBB19 and DBB82 to DBB91 are not important if:
  DB31, ... DBX60.0 (axis / no spindle) = 0
• Axis mode can be activated in all gear steps. If the actual position value encoder is installed on the motor (indirect measuring system), the positioning and contouring accuracy can vary for the different gear steps.

• The gear step cannot be changed when the axis mode is active. The spindle must be switched to control mode. This is done with M41 ... M45 and M5, SPCOF.

• In axis mode, the first parameter set is active (machine data index = zero).

References
Function Manual, Basic Functions; Section "Velocities, setpoint / actual value systems, closed-loop control (G2)" > "Closed-loop control" > "Parameter sets of the position controller"

Dynamic response
The dynamic limits of the axis apply in axis mode. For example:
• MD32000 $MA_MAX_AX_VELO[<axis>] (maximum axis velocity)
• MD32300 $MA_MAX_AX_ACCEL[<axis>] (maximum axis acceleration)
• MD32431 $MA_MAX_AX_JERK[<axis>] (maximum axial jerk for path motion)

Feedforward control
The feedforward control mode active for the axis is retained.
A detailed description of the "Dynamic feedforward control" function can be found in:

References
Function Manual, Extended Functions; Section "Compensations (K3)" > "Dynamic feedforward control (following error compensation)"

Example: Resolution switchover for analog actuator

Switching to axis mode

<table>
<thead>
<tr>
<th>Programming</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPOS=...</td>
<td></td>
</tr>
<tr>
<td>M5</td>
<td>; Controller enable off (by PLC)</td>
</tr>
<tr>
<td></td>
<td>→ is output on PLC</td>
</tr>
<tr>
<td>M70</td>
<td>; Switch actuator (by PLC on account of M70)</td>
</tr>
<tr>
<td></td>
<td>Controller enable on (by PLC)</td>
</tr>
<tr>
<td>C=...</td>
<td>; NC traverses with axis parameter set</td>
</tr>
</tbody>
</table>

Switching to spindle mode

<table>
<thead>
<tr>
<th>Programming</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>C=...</td>
<td></td>
</tr>
</tbody>
</table>
Change to spindle mode

The appropriate parameter set 1 ... 5 is selected for the active gear stage.

Except for tapping with compensating chuck, the feedforward control is switched on when the following applies:
MD32620 $MA_FFW_MODE (feedforward control type) ≠ 0

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>Axis mode</th>
<th>Spindle mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Valid</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>Valid</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>Valid</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>Valid</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>Valid</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>Valid</td>
</tr>
</tbody>
</table>

Spindle mode: Parameter set according to the gear stage

16.2.6.2 Implicit transition to axis mode

Function

The control system detects the transition to axis mode automatically from the program sequence and generates the requisite M70 sequence within the control system. The situation will dictate which steps are performed. At most, these will include:

1. Stopping the spindle
2. Switching on of the position control, treatment of feedforward control, and parameter block changeover
3. Position synchronization of the block preparation (internal preprocessing stop, if necessary)

This function is always active. Explicit programming of M70 in the part program is, therefore, essentially not necessary.
Procedure

Sequence of the implicit transition to axis mode (M70 is not programmed in the part program):

- **Transition from speed control mode (M3, M4, M5, SPCOF, ...) to axis mode:**
  The transition is detected internally by the control, and an intermediate block is inserted in front of the block which requests the axis mode. The block created contains the M70 functionality. The execution duration for this block is more or less the same as the time required to execute a programmed M70 block. Differences may arise in the event of short switchovers when the spindle is stationary (no braking time) if the implicit generation and output of the auxiliary function M70 to the PLC is dispensed with (see MD35035).

- **Transition from positioning mode (M19, SPOS, SPOSA) to axis mode:**
  The transition is executed immediately and without the generation of an intermediate block. Configured accordingly (see MD35035), the auxiliary function M70, which is generated implicitly, is output to the PLC when the block in which the spindle has its axis mode is loaded.

Output of auxiliary functions to PLC

The implicit transition to axis mode can be notified to the PLC in the form of an auxiliary function output.

**Activation/Deactivation**

The activation/deactivation of this functionality is done using machine data:

**MD35035 $MA_SPIND_FUNCTION_MASK (spindle functions)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0</td>
<td>No auxiliary function output to the PLC in the case of M70 functionality which is generated inside the control.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>In the case of M70 functionality which is generated inside the control, the auxiliary function M70 is generated and output to the PLC. The address extension corresponds to the spindle number.</td>
</tr>
</tbody>
</table>

**Note**
An auxiliary function M70 which is programmed in the part program is always output to the PLC.

**Properties**

The properties of the implicitly generated auxiliary function output M70 are "Quick" and "Output during motion". These properties are fixed settings and are independent of the M70 configuration in the auxiliary-function-specific machine data (MD$_{X}$...$_{AUXFU}_{X}$...).

M70 is only generated once during transition to axis mode. No further M70 auxiliary functions are generated and output in adjacent blocks in which the spindle is operated as an axis. M70 is not implicitly generated and output again until axis mode is exited via, for example, SPOS, M3, M4, M5, SPCOF, etc. and following a renewed transition to axis mode.

**Constraints**

**Synchronized actions**
When the spindle is programmed as an axis in synchronized actions, it is essential to continue making provisions in the application to ensure there are criteria for the transition to axis mode.

If the spindle is in speed control mode, the instruction M70 or SPOS must be programmed prior to programming as an axis. Otherwise alarm signals occur during axis programming.

**FC 18**

As with synchronized actions, transition to axis mode must also be undertaken on the application side in FC 18, e.g. through preparatory positioning instructions. Otherwise, the FC 18 call is acknowledged with an error bit in the FC 18 status word.

No auxiliary function M70 is implicitly generated in the event of transition to axis mode through programming via FC 18.

### Examples

#### Example 1:

**Part program: Transition from rotating spindle to axis mode**

Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N05 M3 S1000</td>
<td></td>
</tr>
<tr>
<td>N10 ...</td>
<td></td>
</tr>
<tr>
<td>N15 POS[C]=77</td>
<td>Before loading N15, an M70 intermediate block is generated in which the spindle is stopped, and M70 is output to the PLC.</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

#### Example 2:

**Part program: Transition from positioning mode to axis mode**

Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N05 SPOS=0</td>
<td></td>
</tr>
<tr>
<td>N10 ...</td>
<td></td>
</tr>
<tr>
<td>N15 C77</td>
<td>Output of the implicit M70 to the PLC, no intermediate block.</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

#### Example 3:

**Synchronized actions: Transition from spindle positioning mode to axis mode**

Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHEN COND1==TRUE DO SPOS=180</td>
<td></td>
</tr>
<tr>
<td>WHEN COND2==TRUE DO POS[C]=270</td>
<td>Output of the implicit M70 to the PLC.</td>
</tr>
</tbody>
</table>
Example 4:
Synchronized actions: Transition from speed control mode to axis mode with M70
Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHEN COND1==TRUE DO M3 S1000</td>
<td></td>
</tr>
<tr>
<td>WHEN COND2==TRUE DO M70</td>
<td>Output of M70 to the PLC.</td>
</tr>
<tr>
<td>WHEN COND3==TRUE DO POS[C]=270</td>
<td>No generation of an implicit M70 because axis mode already exists.</td>
</tr>
</tbody>
</table>

Example 5:
Synchronized actions: Invalid transition from speed control mode to axis mode
Configuration: MD35035 $MA_SPIND_FUNCTION_MASK, bit 20 = 1

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHEN COND1==TRUE DO M3 S1000</td>
<td></td>
</tr>
<tr>
<td>WHEN COND2==TRUE DO POS[C]=270</td>
<td>Alarm 20141!</td>
</tr>
</tbody>
</table>

16.2.7 Initial spindle state

Spindle basic setting
The following machine data is used to specify a spindle mode as basic setting:

MD35020 $MA_SPIND_DEFAULT_MODE

<table>
<thead>
<tr>
<th>Value</th>
<th>Spindle basic setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Speed control mode, position control deselected</td>
</tr>
<tr>
<td>1</td>
<td>Speed control mode, position control activated</td>
</tr>
<tr>
<td>2</td>
<td>Positioning mode</td>
</tr>
<tr>
<td>3</td>
<td>Axis mode</td>
</tr>
</tbody>
</table>

Time when the spindle basic setting takes effect
The time when the spindle basic setting takes effect is set in the machine data:

MD35030 $MA_SPIND_DEFAULT_ACT_MASK

<table>
<thead>
<tr>
<th>Value</th>
<th>Effective time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>POWER ON</td>
</tr>
<tr>
<td>1</td>
<td>POWER ON and program start</td>
</tr>
<tr>
<td>2</td>
<td>POWER ON and RESET (M2 / M30)</td>
</tr>
</tbody>
</table>
16.3 **Reference / synchronize**

**Why synchronize?**

In order to ensure that the controller detects the exact position of the spindle when it is switched on, the controller must be synchronized with the position measuring system of the spindle.

The following functions are possible only with a synchronized spindle:

- Thread cutting
- Tapping without compensating chuck
- Axis programming

For further explanations about synchronization of the spindle, see Section "R1: Referencing (Page 1149)".

**Why reference?**

In order to ensure that the controller detects the exact machine zero when it is switched on, the controller must be synchronized with the position measurement system of the rotary axis. This process is known as referencing. The sequence of operations required to reference an axis is known as search for reference.

Only a referenced axis can approach a programmed position accurately on the machine.

For further explanations about referencing the rotary axis, see Section "R1: Referencing (Page 1149)".

**Installation position of the position measurement system**

The position measurement systems can be installed as follows:

- Directly on the motor in combination with a Bero proximity switch on the spindle as a zero-mark encoder
- On the motor via a measuring gearbox in combination with a Bero proximity switch on the spindle as a zero-mark encoder
- Directly on the spindle
- On the spindle via a measuring gearbox in combination with a Bero proximity switch on the spindle as a zero-mark encoder (only with ratios not equal to 1:1)

Where two position measuring systems are provided, they can be installed either in the same location or separately.
Synchronization procedure

When the spindle is switched on, it can be synchronized as follows:

- The spindle is started with a spindle speed ($S$ value) and a spindle rotation ($M3$ or $M4$) and synchronized with the next zero mark of the position measurement system or with the next Bero signal.

- The spindle is to be positioned from standstill using $SPOS$, $M19$ or $SPOSA$. The spindle synchronizes with the next zero mark of the position measurement system or with the next Bero signal. It is then positioned to the programmed position.

- The spindle can be synchronized from the motion (after $M3$ or $M4$) using $SPOS$, $M19$ or $SPOSA$.

The responses are as follows:

- With $SPOS=\langle Pos\rangle$, $SPOS=DC(\langle Pos\rangle)$ and $SPOS=AC(\langle Pos\rangle)$, the direction of motion is retained and the position is approached.

- With $SPOS = ACN(\langle Pos\rangle)$ or $SPOS = ACP(\langle Pos\rangle)$, the position is always approached with a negative or positive direction of motion. If necessary, the direction of motion is inverted prior to positioning.

- Crossing the zero mark in JOG mode by means of direction keys in speed control mode.

Note

It does not make any difference whether the synchronization procedure is initiated from the part program, FC 18 or synchronized actions.

Note

During synchronization of the spindle, all four possible reference point values are effective depending on the measuring system selected. The measurement system offset has the same effect.

The following machine data must be observed:

- MD34080 $MA\_REFP\_MOVE\_DIST$
  (Reference point distance / destination point for a distance-coded system)

- MD34090 $MA\_REFP\_MOVE\_DIST\_CORR$
  (Reference point offset / absolute offset, distance-coded)

- MD34100 $MA\_REFP\_SET\_POS$
  (Reference point value, with distance-coded system without any significance)

If a non-referenced spindle with $SPOS=IC(...)$ and a path $< 360$ degrees is positioned, it may be the case that the zero mark is not crossed and the spindle position is still not synchronized with the zero mark. This can happen:

- After POWER ON
- By setting the axial NC/PLC interface signals:
  DB31, ..., DBX17.5 (resynchronize spindle when positioning 2)
  DB31, ..., DBX17.4 (resynchronize spindle when positioning 1)

Special features for synchronization with BERO

The position falsification caused by the signal delay with BERO can be corrected internally in the NC by entering a signal runtime compensation.
The signal runtime compensation is set by means of the machine data:

- MD31122 $MA_BERO_DELAY_TIME_PLUS
  (BERO delay time for a positive direction of motion)
- MD31123 $MA_BERO_DELAY_TIME_MINUS
  (BERO delay time for a negative direction of motion)

The effect depends on the setting in machine data:

MD34200 $MA_ENC_REFP_MODE (referencing mode)

MD34200 = 7

The setting MD34200 $MA_ENC_REFP_MODE = 7 only executes position synchronization at a velocity/speed which is fixed in machine data:

MD34040 $MA_REFP_VELO_SEARCH_MARKER (reduced velocity)

The zero mark is not automatically sought, it has to be requested explicitly with the 0-1 edge of the NC/PLC interface signal:

DB31, ... DBX16.4/5 (resynchronize spindle 1/2*)

*) 1/2 stands for the selected measuring system.

The velocity defined in MD34040 is also effective when referencing in JOG-REF mode and through the part program with G74.

MD34200 = 2

Setting MD34200 $MA_ENC_REFP_MODE = 2 executes position synchronization without specifying a specific velocity/speed.

Note

Signal propagation delays are preset on delivery so that the content generally does not have to be changed.

Referencing sequence

If the spindle is to be programmed in axis mode directly after controller power-up, it must be ensured that the axis is referenced.

When the controller is switched on, the spindle can be referenced (condition is one zero mark per revolution).

For information about the referencing procedure, see Section "R1: Referencing (Page 1149)".

The rotary axis is referenced at the same time as the spindle is synchronized (see section "Synchronization procedure") if the position measuring system used for the spindle is also used for the rotary axis.

Position measurement systems, spindle

The spindle can be switched from spindle mode to axis mode (rotary axis) if a single motor is used for spindle mode and axis mode.

The spindle (spindle mode and axis mode) can be equipped with one or two position measurement systems. With two position measurement systems, it is possible to assign one position measurement system to the spindle and the other to the rotary axis, or to assign two position measurement systems to the spindle. Where two position measurement systems are provided, both are updated by the controller, but only one can be active.
The active position measuring system is selected using the NC/PLC interface signal:
DB31, ... DBX1.5 (position measuring system 1)
or
DB31, ... DBX1.6 (position measuring system 2)
The active position measurement system is required for the following functions:

- Position control of the spindle (SPCON)
- Spindle positioning (SPOS, M19 and SPOSA)
- Thread cutting (G33, G34, G35)
- Tapping without compensating chuck (G331, G332)
- Revolutionary feedrate (G95)
- Constant cutting rate (G96, G961, G97, G971)
- Spindle actual speed display
- Axis mode
- Synchronous spindle setpoint coupling

Resynchronizing the position measuring system for the spindle

In the following cases, the spindle position measurement system must be resynchronized:

- The position encoder is on the motor, a Bero proximity switch is mounted on the spindle and a gear stage change is performed. Synchronization is triggered internally once the spindle is rotating in the new gear stage (see Synchronization procedure).

- The machine has a selector switch for a vertical and horizontal spindle. Two different position encoders are used (one for the vertical spindle and one for the horizontal spindle), but only one actual value input is used on the controller. When the system switches from the vertical to the horizontal spindle, the spindle must be resynchronized. This synchronization is initiated with the NC/PLC interface signal:
DB31, ... DBX16.4 (resynchronize spindle 1)
or
DB31, ... DBX16.5 (resynchronize spindle 2)
The spindle must be in open-loop control mode.

Position restoration with POWER ON

For spindles with incremental position measuring systems, it is possible to buffer the actual values after a POWER OFF and after POWER ON, to restore the position last buffered before switching-off, in order that position-dependent functions, e.g. transformation can be restored (see Section "Automatic restoration of the machine reference (Page 1190)"). One application is, e.g. tool retraction after POWER OFF when machining with tool orientation (see Section "Tool withdrawal after POWER ON with orientation transformation (Page 614)").

The following NC/PLC interface signals display the state of the position measuring system after position restoration:
DB31, ... DBX71.4 ("Restored 1") for position measuring system 1
DB31, ... DBX71.5 ("Restored 2") for position measuring system 2
Once the tool has been retracted in the JOG mode, axes whose positions have been restored are referenced. As a consequence, signals DB31, ... DBX71.4/5 ("Restored 1/2") are deleted and signals DB31, ... DBX60.4/5 ("Referenced/synchronized 1/2") are set.

**Note**

If machine data MD20700 $MC_REFP_NC_START_LOCK is set to a value of "2", then an NC start is also possible with "restored" axis positions (in the MDA mode or when overstoring).

---

### 16.4 Configurable gear adaptation

#### 16.4.1 Gear stages for spindles and gear change change

**Why do we need gear stages?**

Gear stages are used on spindles to step down the speed of the motor in order to generate a high torque at low spindle speeds or to step up in order to maintain a high speed.

**No. of gear stages**

Five gear stages can be configured for each spindle.

The number of used gear stages is defined in machine data:

MD35090 $MA_NUM_GEAR_STEPS

**Parameterization of the gear stages**

The gear stages 1 to 5 can be parameterized via the following machine data:

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD35012 $MA_GEAR_STEP_CHANGE_POSITION[&lt;n&gt;]</td>
<td>Gear stage change position</td>
</tr>
<tr>
<td>MD35110 $MA_GEAR_STEP_MAX VELO[&lt;n&gt;]</td>
<td>Maximum speed for automatic gear stage change</td>
</tr>
<tr>
<td>MD35120 $MA_GEAR_STEP_MIN VELO[&lt;n&gt;]</td>
<td>Minimum speed for automatic gear stage change</td>
</tr>
<tr>
<td>MD35130 $MA_GEAR_STEP_MAX VELO LIMIT[&lt;n&gt;]</td>
<td>Maximum speed of gear stage</td>
</tr>
<tr>
<td>MD35135 $MA_GEAR_STEP_PC MAX VELO LIMIT[&lt;n&gt;]</td>
<td>Maximum speed of gear stage in position control</td>
</tr>
<tr>
<td>MD35140 $MA_GEAR_STEP_MIN VELO LIMIT[&lt;n&gt;]</td>
<td>Minimum speed of gear stage</td>
</tr>
<tr>
<td>MD35200 $MA_GEAR_STEP_SPEEDCTRL ACCEL[&lt;n&gt;]</td>
<td>Acceleration in speed control mode</td>
</tr>
<tr>
<td>MD35210 $MA_GEAR_STEP_POSCTRL ACCEL[&lt;n&gt;]</td>
<td>Acceleration in position control mode</td>
</tr>
<tr>
<td>MD35300 $MA_SPIND_POSCTRL VELO[&lt;n&gt;]</td>
<td>Position control activation speed</td>
</tr>
</tbody>
</table>
### Type of gear stage change

The type of gear stage change is set in machine data:

**MD35010 $MA_GEAR_STEP_CHANGE_ENABLE**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>The spindle motor is attached to the spindle directly (1:1) or with a non-variable transmission ratio (basic setting). The machine data of the first gear stage is effective.</td>
</tr>
</tbody>
</table>
| 1   | Spindle motor with up to five gear stages. The gear stage change takes place:  
• In oscillation mode  
• At indefinite change position |
| 0   | Meaning as in Bit 0 = 0. |
| 1   | Meaning as in Bit 0 = 1, however, the gear stage change takes place at the configured spindle position. The change position is set in machine data:  
**MD35012 $MA_GEAR_STEP_CHANGE_POSITION**  
The position is approached in the current gear stage before the gear stage change. If Bit 1 is set, then Bit 0 is ignored! |
| 3   | The gear stage change dialog between NCK and PLC is simulated. |
| 5   | The second gear stage data set is used while tapping with G331/G332 (see the following paragraph "Second gear stage data set"). The bit must be set for the master spindle used during the tapping. |

### Requirement for a gear stage change

In principle, the gear stage change is only performed if the requested gear stage is not the same as the active gear stage.

### Parameter set selection during gear stage change

The servo parameter set is also changed over with the gear stage if:

**MD35590 $MA_PARAMSET_CHANGE_ENABLE = 0 or 1**

For further information, see Section "Parameter set selection during gear step change (Page 1246)".
Request gear stage change

A gear stage change can be requested:

- **In the part program using:**
  - M40 S...
    Automatic gear stage selection to the programmed speed S...
  - M41 ... M45
    Direct selection of gear stages 1 ... 5
  - M70
    For MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE = 1 ... 5
    (see "Configurable gear step in M70 (Page 1263")
  - G331 S...
    For MD35010 $MA_GEAR_STEP_CHANGE_ENABLE, Bit 5 = 1

- **In synchronized actions using:**
  - DO M40 S...
    Automatic gear stage selection to the programmed speed S...
  - DO M41... M45
    Direct selection of gear stages 1 ... 5
  - DO M70
    For MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE = 1 ... 5

- **Through the PLC using the FC18 function block**

- **In the reset state through description of NC/PLC interface:**
  DB31, ... DBX16.0-16.2 (actual gear stage A to C)
  The mechanically active gear stage can be communicated to the NC especially after a POWER ON.

**Note**

If the spindle motor is attached to the spindle directly (1:1) or with a non-variable transmission ratio (MD35010 = 0), then the M40 and M41 ... M45 auxiliary functions are not relevant to this spindle.
Gear stage change

Gear stage selection between two gear stages with specification of a maximum spindle speed is shown in the example below:

![Diagram of gear stage change with selection between two gear stages](image)

In the case of automatic gear step selection, this speed range is not utilized to full advantage in gear step 1.

Specifiable using MD35110 GEAR_STEP_MAX VELO:

- \( n_{1\text{max}} \) ... Maximum spindle speed for gear step 1
- \( g_{1\text{min}} \) ... Minimum spindle speed for gear step 1, automatic gear step selection
- \( g_{1\text{max}} \) ... Maximum spindle speed for gear step 1, automatic gear step selection
- \( n_{2\text{max}} \) ... Maximum spindle speed for gear step 2
- \( g_{2\text{min}} \) ... Minimum spindle speed for gear step 2, automatic gear step selection
- \( g_{2\text{max}} \) ... Maximum spindle speed for gear step 2, automatic gear step selection

Figure 16-3  Gear stage change with selection between two gear stages
Process sequence of the gear stage change

If the new gear stage is preselected, the following sequence is implemented:

1. Changeover sequence

   The two following NC/PLC interface signals are set:
   DB31, ... DBX82.0-82.2 (setpoint gear stage A to C)
   DB31, ... DBX82.3 (change over gear stage)

   In accordance with the point at which NC/PLC IS:
   DB31, ... DBX18.5 (oscillation speed)
   is set, the spindle decelerates to a standstill at the acceleration for oscillation or at the
   acceleration for speed control / position control.

   Oscillation can be activated at the latest when the spindle reaches a standstill:
   DB31, ... DBX61.4 (axis/spindle stationary)
   with NC/PLC IS:
   DB31, ... DBX18.5 (oscillation speed).

   In principle, the new gear stage can also be engaged without oscillation

   When the new gear stage is engaged, the following NC/PLC interface signals are set by
   the PLC program:
   DB31, ... DBX16.0-16.2 (actual gear stage A to C)
   DB31, ... DBX16.3 (gear is changed)

2. End of gear stage change

   The gear stage change is treated as completed (spindle operation type "oscillation mode"
   is deselected), if the following NC/PLC interface signal is set:
   DB31, ... DBX16.3 (gear is changed)

   The new actual gear stage is changed to the servo and interpolation parameter set when
   the motor is stationary.

   With NC/PLC interface signal:
   DB31, ... DBX16.3 (gear is changed)
   is used to communicate to the NC that the new gear stage is valid and the oscillation mode
   can be completed.

   NC/PLC IS:
   DB31, ... DBX82.3 (change gear)
   is reset by the NCK,
   which causes the PLC program to reset NC/PLC IS:
   DB31, ... DBX16.3 (gear changed).

   In this case, it does not matter whether NC/PLC IS:
   DB31, ... DBX18.5 (oscillation mode)
   is still set.

   The actual gear stage, which should correspond to the set gear stage, is relevant
   for selecting the parameter set.

   If this is not the case, then Alarm 22010 :
   MD11410 $MN_SUPPRESS_ALARM_MASK, Bit 3 = 0
   is output.

   Following acknowledgement of gear stage change via the PLC
   with NC/PLC IS:
   DB31, ... DBX16.3 (gear changed)
   the spindle is in speed control mode (DB31, ... DBX84.7).

   For further information on the signal exchange between PLC and NC, see Section "A2: Various NC/PLC interface signals and functions (Page 35)".
Second gear stage data set

The automatic gear stage change M40 can be extended by a second configurable gear stage data set.

The second gear stage data set is used exclusively in connection with tapping without compensation chuck (G331, G332) so that an effective adjustment of spindle speed and motor torque can be achieved.

The activation is undertaken by setting the following bit for the master spindle:

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE, Bit 5 = 1

The number of used gear stages of the second gear stage data set is defined with the machine data:

MD35092 $MA_NUM_GEAR_STEPS2

The second gear stage block data set is deactivated if:

MD35092 $MA_NUM_GEAR_STEPS2 = 0 (basic setting)

The first gear stage data set then selects the gear stage when M40 is active.

Note

The number of gear stages in the second data set can vary from the first. If no appropriate gear stage is found for a programmed speed for M40, then no gear stage change is carried out (exceptions, see "M40: Automatic gear stage selection for speeds outside the configured switching thresholds (Page 1293)").

For more information about a typical program sequence in thread cutting without compensating chuck G331/G332 see:

References:
Programming Manual - Fundamentals; Motion Commands

The gear stages 1 to 5 of the second gear stage data set can be parameterized via the following machine data:

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD35112 $MA_GEAR_STEP_MAX_VELO2[n]</td>
<td>Maximum speed for automatic gear stage change</td>
</tr>
<tr>
<td>MD35122 $MA_GEAR_STEP_MIN_VELO2[n]</td>
<td>Minimum speed for automatic gear stage change</td>
</tr>
<tr>
<td>MD35212 $MA_GEAR_STEP_POSCTRL_ACCEL2[n]</td>
<td>Acceleration in position control mode</td>
</tr>
</tbody>
</table>

Note

The number of servo parameter sets concerning the mechanical factors remain unchanged. Furthermore, five mechanical gear stages for the spindle and one for the axis operation can be configured.
The speed limitations are configured only once for each gear stage with the following machine data, independently of the different switching thresholds:

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[n]</td>
<td>Maximum speed of gear stage</td>
</tr>
<tr>
<td>MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT[n]</td>
<td>Minimum speed of gear stage</td>
</tr>
</tbody>
</table>

For tapping without compensating chuck (G331, G332) the speed can be limited to the linear acceleration range of the motor additionally. For this, the maximum speed of the linear motor characteristics range is specified in the following machine data as a function of the gear stage:

MD35550 $MA_DRILL_VELO_LIMIT[n]

Specify gear stage in the part program

Automatic selection with active M40

The gear stage is automatically selected by the control. The gear stage in which the programmed spindle speed ($...$) is possible is checked in this context. If a gear stage results from this that is not equal to the current (actual) gear stage, then the following NC/PLC interface signals are set:

DB31, ... DBX82.3 (change over gear stage)
DB31, ... DBX82.0-82.2 (setpoint gear stage A to C)

While the appropriate gear stage is being determined, a gear stage change is only requested if the new speed is not within the permissible speed range of the active gear stage.

The speed is limited to the maximum speed of the current gear stage or raised to the minimum speed of the current gear stage and the appropriate NC/PLC interface signal is set:

DB31, ... DBX83.1 (speed setpoint limited)
DB31, ... DBX83.2 (speed setpoint increased)
16.4 Configurable gear adaptation

Figure 16-4  Example for two gear stages with overlapping speed ranges for automatic gear stage change (M40)

Note
In the case of M40, the spindle must be in open-loop control mode for automatic gear stage selection with an S word. Otherwise the gear stage change is rejected and the following alarm is set:

Alarm 22000 "gear stage change is not possible"

Note
An active reduction gear is not considered in the selection for the automatic gear stage change.

Permanently defining the gear stage with M41 to M45

The gear stage can be permanently defined in the part program with M41 to M45.

If a gear stage is specified via M41 to M45 that is not equal to the current (actual) gear stage, then the following NC/PLC interface signals are set:

DB31, ... DBX82.3 (change over gear stage)
DB31, ... DBX82.0-82.2 (setpoint gear stage A to C)
The programmed spindle speed \( (S...) \) then refers to this permanently defined gear stage:

- If a spindle speed is programmed and it is higher than the maximum speed of the permanently defined gear stage (MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT), then the speed is decreased to this limit and the following NC/PLC interface signal is set: DB31, ... DBX83.1 (speed setpoint limited)
- If a spindle speed is programmed and it is lower than the minimum speed of the permanently defined gear stage (MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT), then the speed is increased to this minimum and the following NC/PLC interface signal is set: DB31, ... DBX83.2 (speed setpoint increased)

**Block change**

When programming the gear stage change in the part program, the gear stage change set remains active until it is aborted by PLC.

This corresponds to the effect as if the following NC/PLC interface signal were set:
- DB21, ... DBX6.1 (read-in disable)

**Specification of gear stage via PLC with FC18**

The gear stage change can also be performed by function block FC18 during a part program, in the reset state or in all operating modes.

If the speed and direction of rotation is specified with FC18, the NC can be requested to select the gear stage as appropriate for the speed. This corresponds to an automatic gear stage change with M40.

The gear stage is not changed if:
- The spindle is positioned via FC18.
- The spindle is traversed in the axis mode.

For further information on the FC18 function block, see Section "P3: Basic PLC program for SINUMERIK 840D sl (Page 821)".

**Specification of a gear stage in synchronized actions**

The gear stage change can be requested by synchronized actions using:

- **DO M40 S...**
  Automatic gear stage selection to the programmed speed \( S... \)

- **DO M41... M45**
  Direct selection of gear stages 1 ... 5

- **DO M70**
  For MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE = 1 ... 5
  (see "Configurable gear step in M70 (Page 1263)"
The gear stage is **not** changed if:

- The spindle is positioned via synchronized actions.
- The spindle is traversed in the axis mode.

**Note**

For further details, please refer to the section "Specification of a gear stage in part program".

**Exception:**

The block change is not affected by the specification of a gear stage in synchronized actions.

---

**Manual specification of a gear stage**

Outside a part program that is running, the gear stage can also be changed without a request from the NC or the machine. This is the case, for example, when a gear stage is changed manually.

To select the appropriate parameter set, the NC must be informed of the current gear stage. To enable this, the control or the part program must be in the reset state.

**Supplementary conditions**

Transfer of the gear stage to the NC is initiated when NC/PLC IS:

DB31, ... DBX16.0-16.2 (actual gear stage A to C) changes.

These three bits must be set continuously during operation.

Successful transfer is acknowledged with NC/PLC IS:

DB31, ... DBX82.0-82.2 (set gear stage A to C)

to the PLC.

NC/PLC IS:

DB31, ... DBX16.3 (gear changed)

must not be set.

If position control is active when a new gear stage is specified by the PLC with DB31, ... DBX16.0-16.2, then it is switched off for the duration of this changeover sequence.
NC stop during gear stage change

The spindle cannot be stopped with NC/PLC IS:
DB21, ... DBX7.4 (NC stop)
if:

- The spindle is not yet in oscillation mode for the gear stage change.
- NC/PLC IS:
  DB31, ... DBX16.3 (gear changed)
  is not set.

Note
Options for aborting:
DB31, ... DBX2.2 (delete distance-to-go / spindle reset)
or
DB31, ... DBX16.3 (gear changed)
with corresponding acknowledgement from actual gear stage:
DB31, ... DBX16.0-16.2 (actual gear stage).

Spindle response after a gear stage change

How the spindle behaves once the gear stage has been changed depends on the following initial conditions:

- If the spindle was in the stop state before the gear stage change (M5, FC18: "Stop rotate spindle"), in positioning or axis mode, M5 (spindle stop) is active after completion of the gear stage change.
- If a direction of rotation (M3, M4, FC18: "Start spindle rotation"), then the last speed and direction of rotation will become active again after the gear stage change. In the new gear stage, the spindle accelerates to the last spindle speed programmed ($...$).
- If position control was active before the gear stage change (SPCON), then it is reactivated after the gear stage change.
  The next block in the part program can be executed.
Special features

The following points must be observed on gear stage change:

- The gear stage change is not terminated by selecting
  NC/PLC IS:
  DB31, ... DBX20.1 (run-up switchover to V/f mode).
  Setpoint 0 is output.
  The gear stage change is acknowledged as usual
  via the NC/PLC interface signal:
  DB31, ... DBX16.3 (gear is changed)
- The "Ramp-function generator rapid stop" signal must be reset by the PLC before the gear
  stage change is completed by the PLC.
- The process sequence of the gear stage change is ended during NC reset without any
  alarm output.
  The gear stage output with NC/PLC IS:
  DB31, ... DBX16.0-16.2 (actual gear stage A to C)
  is applied by the NC.

Star/delta switchover with FC17

Digital main spindle drives can be switched in both directions between star and delta using
FC17, even when the spindle is running. This automatic switchover is controlled by a defined
logic circuit in FC17 which provides the user with a configurable switchover time for the relevant
spindle.

For further information on the FC17 function block, see Section "P3: Basic PLC program for
SINUMERIK 840D sl (Page 821)."

16.4.2 Spindle gear stage 0

Technical background

For machine's where the spindle load gear can be changed over, situations can occur where
the gear train between the motor and load (workpiece/tool) is interrupted. This state can occur,
e.g. when pressing Reset or Emergency Stop while performing a gear stage change or when
the machine is commissioned for the first time while it is being installed. The control must
identify this state where the gear train is open and the next gear stage change request must
be unconditionally executed.

Function

When the gear is disengaged, the binary-coded value "0" (≠ gear stage 0) is transferred to the
NC from the PLC using the interface signal bits DB31, ... DBX16.0-2 (actual gear stage A to
C):
DB31, ... DBX16.0-2 = 0
The value is used by the control to identify the state where the gear train is open.
**Effects on the gear stage change**

**Gear stage change in the part program**

The actual gear stage signaled from the PLC is read by the NC when starting a part program. If, at this instant in time, a value of "0" is read for the actual gear stage, then the next gear stage change is executed and the gear stage change dialog is performed by the PLC. If a value greater than "0" is read, then already in the program a comparison is made between the requested and active gear stage. If both gear stages are the same, the gear stage is not changed and a possibly programmed path motion is not interrupted.

**Gear stage change in synchronized actions, FC18 and DBB30**

The actual gear stage signaled from the PLC is always evaluated by the NC when the gear stage is changed. The gear stage is always changed if a value of "0" is read from the NC. When reading a value greater than "0", a comparison is made between the requested and active gear stage. The gear stage is only changed with the PLC if the two values are not equal and the NC/PLC interface signal DB31, ... DBX82.3 (change over gear) is then output.

**Boundary conditions**

- **Output of DB31, ... DBX16.0-2 = 0**
  When the gear is disengaged, the PLC must enter gear stage 0 in the NC/PLC interface DB31, ... DBX16.0-2 (actual gear stage A to C).

- **Enabling the gear stage change**
  The precondition for a gear stage change after reaching gear stage 0 is the general enable of the gear stage change via via machine data:
  - MD35010 $MA_GEAR_STEP_CHANGE_ENABLE (assign parameters to the gear stage change)
  - MD35090 $MA_NUM_GEAR_STEPS (number of gear stages set up)
  - MD35092 $MA_NUM_GEAR_STEPS2 (2nd gear stage data set: Number of gear stages that have been created) if MD35010 $MA_GEAR_STEP_CHANGE_ENABLE, bit 5 = 1

- **PLC user program/ POWER ON ASUB**
  The PLC user program or POWER ON ASUB should ensure that when the gear is disengaged (gear stage 0) before spindle motion, a gear stage change request is programmed. For instance, this can be realized with M41 in the ASUB. Spindle motion such as e.g. in JOG or in axis operation does not generate any gear stage change itself.

**Example**

Example for the sequence to select the first gear stage after POWER ON

1. POWER ON.
2. The PLC user program determines, in the mechanical environment, the "Gear is disengaged" state.
3. The PLC transfers the "Gear is disengaged" state to the NC by setting: DB31, ... DBX16.0-2 = 0
4. Part program start or POWER ON ASUB.
5. N05 (part program, refer below) is executed:
The gear stage is changed to gear stage 1.
From the NC:
– the following NC/PLC-interface signal is set:
  DB31, ... DBX82.3 (change over gear stage)
– the setpoint gear stage 1 is signaled to the PLC:
  DB31, ... DBX82.0 = 1
  DB31, ... DBX82.1 = 0
  DB31, ... DBX82.2 = 0

6. Mechanical gear stage change, acknowledgement
If the gear stage is selected, then from the PLC:
– the following NC/PLC-interface signal is set:
  DB31, ... DBX16.3 (gear is changed)
– Actual gear stage 1 signaled to the NC:
  DB31, ... DBX16.0 = 1
  DB31, ... DBX16.1 = 0
  DB31, ... DBX16.2 = 0

7. N80 is executed:
Due to the optimization of the gear stage change frequency in the part program, the gear
stage is not changed.

Part program:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N05 M41</td>
<td>; Select 1st gear stage</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N80 M41</td>
<td>; No gear stage change, if the 1st gear stage is selected.</td>
</tr>
</tbody>
</table>

Configuring data for spindle 1 (AX5):
MD35010 $MA_GEAR_STEP_CHANGE_ENABLE[AX5] = 1 (enable gear stage change)

16.4.3 Determining the spindle gear stage

The actual stage of a spindle can be read using system variables:

- For the display in the user interface, in synchronized actions or with a preprocessing stop in the part program via the system variables:

  $VC_SGEAR[<n>]  
  Currently selected spindle gear stage
  $VC_SGEAR reads the actual gear stage signaled from the PLC.
  Range of values: 0 ... 5
$AC_SGEAR[n]$  
Active spindle gear stage
$AC_SGEAR$ reads the setpoint gear stage in the main run.
Range of values: 1 ... 5
The data set for the spindle is activated corresponding to this gear stage.

**Note**

For a search, the actual gear stage ($VC_SGEAR[n]$) can differ from the setpoint gear stage ($AC_SGEAR[n]$) as, during the search, no gear stage change takes place. Therefore, using $VC_SGEAR[n]$ and $AC_SGEAR[n]$, it can be interrogated whether a gear stage change should be made after a search.

- Without preprocessing stop in the part program via system variables:

$P_SGEAR[n]$  
Setpoint gear stage
$P_SGEAR$ reads the gear stage programmed in the part program (M41 ... M45), for M40 selected, or for M70, the configured gear stage.

$P_SEARCH_SGEAR[n]$  
Search: Gear-specific M function
$P_SEARCH_SGEAR$ contains the last programmed gear stage M function collected with the block search.

### 16.4.4 Parameter set selection during gear step change

**Servo parameter sets**

The servo parameter sets 1 to 6 adapt the position controller to the changed properties of the machine during a gear change of the spindle.

**Parameter set selection during gear stage change**

The gear stage parameter set (interpolation parameters) and, depending on the setting in the following machine data, the servo parameter set are also modified during gear stage change.
MD35590 $MA_PARAMSET_CHANGE_ENABLE (parameter set change possible)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0     | In-system parameter set selection  
The parameter sets of the servo are assigned permanently.  
The following applies:  
• For axes and spindles in the axis mode, the first parameter set is active in principle.  
  Exception:  
  For G33, G34, G35, G331 and G332, for the axes involved, the parameter set with the  
  following number is activated:  
  Master spindle gear stage + 1 (corresponds to parameter set No. 2 ... 6)  
• For spindles in the spindle mode, the parameter set is set matching the gear stage. |
| 1     | Besides the in-system parameter set selection, there is also the option of an "external" parameter set selection.  
• By the PLC (DB31, ... DBX 9.0 - 9.2)  
• Via programming of INTPARA in the part program or in synchronized actions  
However, the in-system parameter set selection has priority.  
Note: Value 1 is relevant only to axes. |
| 2     | The servo parameter set is specified exclusively by the PLC (DB31, ... DBX 9.0 - 9.2) or through the programming of SCPARA in the part program or in synchronized actions (for axes and spindles).  
The 1st parameter set is selected after POWER ON. |

Spindle mode

MD35590 $MA_PARAMSET_CHANGE_ENABLE = 0 or 1
The parameter set is selected according to the gear stage + 1.
The active gear stage is located in:
DB31, ... DBX16.0-16.2 (actual gear stage A to C)
The active parameter set is output in:
DB31, ... DBX69.0-69.2 (controller parameter set A to C)
One set of parameters, with the following assignment, is provided by the NC for each of the five gear stages:

<table>
<thead>
<tr>
<th>Data set for ...</th>
<th>NC/PLC interface</th>
<th>Parameter set</th>
<th>Parameter set index [n]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis mode</td>
<td>DBX 69.2 / 69.1 / 69.0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Gear stage 1</td>
<td>001</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Gear stage 2</td>
<td>010</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Gear stage 3</td>
<td>011</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Gear stage 4</td>
<td>100</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Gear stage 5</td>
<td>101 110 111</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>
Spindle in axis mode

If the spindle is in axis mode, the parameter set index "0" is selected in the servo (note MD35590 $MA_PARAMSET_CHANGE_ENABLE!).

The gear stage change behavior depends on the setting in the machine data:

MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE (gear stage for axis mode in M70)

If there is no gear stage configured for axis mode (MD35014 = 0), no implicit gear stage change takes place in M70 (default setting!). The last gear stage is saved internally and is reactivated with the associated parameter set during the next spindle programming.

If, however, a gear stage is configured for axis mode (MD35014 = 1 ... 5), a gear stage change to gears 1 ... 5 takes place during the execution of M70. When changing from axis mode to spindle mode, the gear stage loaded with M70 remains activated. The gear stage which is activated in spindle mode prior to M70 is not automatically loaded again.

See also "Configurable gear step in M70 (Page 1263)".

Load gearbox transmission ratio

It is possible to configure positive or negative load gearbox factors for each gear stage and in axis mode.

The setting is undertaken separately for numerator and denominator via the machine data:

MD31050 $MA_DRIVE_AX_RATIO_DENOM[n] (load gearbox denominator)
MD31060 $MA_DRIVE_AX_RATIO_DENOM[n] (load gearbox numerator)

The setting range is the same size for positive and negative load gearbox factors.

It is not possible to enter the value "0".

Note

If an indirect encoder is configured, and the load gearbox transmission ratio changes, then the reference is lost and the NC/PLC interface signal:
DB31, ... DBX60.4/60.5 (referenced / synchronized 1 or 2)
is reset for the relevant measuring system.

References

For further information about control and servo parameter set, please refer to:

- Functions Manual - Basic Functions; Velocities, Setpoint-Actual Value Systems, Closed-Loop Control (G2)
- Programming Manual, Job Planning; Section: Programmable servo parameter set
16.4.5 Intermediate gear

Application and functions

A configured intermediate gear can be used to adapt a variety of rotating tools. The intermediate gear on the tool side has a multiplicative effect on the motor/load gearbox.

It is set via machine data:

MD31066 $MA_DRIVE_AX_RATIO2_NUMERA (intermediate gear numerator)
MD31064 $MA_DRIVE_AX_RATIO2_DENOM (intermediate gear denominator)

An encoder on the tool for the intermediate gear is configured with machine data:

MD31044 $MA_ENC_IS_DIRECT2 (encoder on intermediate gear)

Change parameters for this machine data can be activated with "NewConfig" either using the SinuCOM-NC commissioning software or via a softkey on the operator panel (HMI). The existing motor/load gearboxes, on the other hand, are active after POWER ON.

Tool change

If the intermediate gear is changed at the same time as the tool, the user must also reconfigure the transmission ratio of the numerator and denominator via the machine data of the intermediate gear.

Example:

<table>
<thead>
<tr>
<th>CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineering error</td>
</tr>
<tr>
<td>It remains the task of the user to stop within the appropriate period in order to make changes to the machine data when required and then activate a &quot;NewConfig&quot;.</td>
</tr>
</tbody>
</table>

In the case of an installed tool with a transmission ratio of 2:1, a suitable intermediate gear is configured and is activated immediately in the part program with the command NEWCONF.

Program code

```
N05 $MA_DRIVE_AX_RATIO2-NUMERA[AX5] = 2
M10 $MA_DRIVE_AX_RATIO2-DENOM[AX5] = 1
N15 NEWCONF
```
Changeover

Changeover to a new transmission ratio is performed immediately by means of NewConfig. From a technological viewpoint, the associated mechanical changeover process takes some time, since, in mechanical terms, a different intermediate gear with rotating tool is being installed.

**Note**

At zero speed, changeover is jerk-free. The user is therefore responsible for taking appropriate precautions.

Applications in which changeover takes place during motion and which require smoothed or soft speed transition can be handled using existing setpoint speed filters.

For further explanations regarding control engineering dependencies, see Section "G2: Velocities, setpoint / actual value systems, closed-loop control (Page 317)."

16.4.6 Nonacknowledged gear step change

Mode change

A gear stage change that has not been acknowledged cannot be interrupted by a change in operating mode (e.g. switchover to JOG).

The switchover is delayed by the maximum period entered in machine data:

MD10192 $MN_GEAR_CHANGE_WAIT_TIME

If the gear stage change is not acknowledged within this time, the NC will output an alarm:

Further events

Events that initiate reorganization will also wait until a gear stage change is completed.

The time entered in machine data:

MD10192 $MN_GEAR_CHANGE_WAIT_TIME
determines how long the control waits before executing the gear stage change. If this time elapses without the gear stage change being completed, the NC responds with an alarm.

The following events have an analog response:

- User ASUB
- Mode change
- Delete distance-to-go
- Axis interchange
- Activate PI user data
- Enable PI service machine data
Switch over skip block, switch over Dry Run
 Editing in the modes
 Compensation block alarms
 Overstore
 Rapid retraction with G33, G34, G35
 Subprogram level abort, subprogram abort

Response after POWER ON

The active gear stage on the machine can be specified by the PLC after POWER ON and in the RESET state.

The NCK will then select the appropriate parameter set and check back the NC/PLC interface signals: DB31, ... DBX82.0-82.2 (set gear stage A to C) to the PLC.

16.4.7 Gear step change with oscillation mode

What is oscillation?

Oscillation in this context means that the spindle motor rotates alternately in the clockwise and counter-clockwise directions. This oscillation movement makes it easy to engage a new gear stage.

Oscillation mode

NC/PLC IS:
DB31, ... DBX82.3 (change gear)
displays that a gear stage change is required.

In principle, the new gear stage can also be engaged without oscillation

1. MD35010 $MA_GEAR_STEP_CHANGE_ENABLE must be set to 1.
2. NC/PLC interface signal DB31, ... DBX84.6 (oscillation mode) is set.
3. The acceleration is set in the machine data:
   MD35410 $MA_SPIND_OSCILL_ACCEL

DB31, ... DBX18.5 (oscillation speed)

The spindle is in oscillation mode if a new gear stage was defined using automatic gear stage selection (M40) or M41 to M45 (DB31, ... DBX82.3 (change gear) is set).

NC/PLC IS:
DB31, ... DBX82.3 (change gear)
is only enabled when a new gear stage is defined
that is not the same as the current actual gear stage.

If NC/PLC IS:
DB31, ... DBX18.5 (oscillation speed)
is simply set by the PLC without a new gear stage being defined by the NC,
the spindle does not change to oscillation mode.

Oscillation mode is activated with NC/PLC IS:
DB31, ... DBX18.5 (oscillation speed)

Depending on NC/PLC IS:
DB31, ... DBX18.4 (oscillation via PLC)
while the function is in operation, a distinction is made between:

- Oscillation via NCK
- Oscillation via PLC
- Oscillation with FC 18

References:
Function Manual, Basic Functions; PLC Basic Program (P3)

Oscillation time

The oscillation time for oscillation mode can be defined in a machine date for each direction
of rotation:

<table>
<thead>
<tr>
<th>Oscillation time in M3 direction (referred to as t1 in the following):</th>
<th>MD35440 $MA_SPIND_OSCILL_TIME_CW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oscillation time in M4 direction (referred to as t2 in the following):</td>
<td>MD35450 $MA_SPIND_OSCILL_TIME_CCW</td>
</tr>
</tbody>
</table>

Oscillation via NCK

Phase 1:

NC/PLC IS:
DB31, ... DBX18.5 (oscillation speed)
accelerates the spindle motor to the speed (with oscillation acceleration) defined in machine
data:
MD35400 $MA_SPIND_OSCILL_DES_VELO (oscillation speed)

Start direction is defined through the following machine data:
MD35430 $MA_SPIND_OSCILL_START_DIR (start direction with oscillation)

The time t1 (or t2) is started
according to which start direction is given in the machine data:
MD35430 $MA_SPIND_OSCILL_START_DIR

The time - and not the fact that the oscillation speed is reached - is always decisive.

Phase 2:
If time $t_1$ ($t_2$) has elapsed, the spindle motor accelerates in the opposite direction to the speed defined in machine data: 

$$ \text{MD35400 $\text{MA\_SPIND\_OSCILL\_DES\_VELO}$} \ . $$

$\text{Time } t_2$ ($t_1$) starts.

**Phase 3:**

If time $t_2$ ($t_1$) has passed, the spindle motor accelerates in the opposite direction (same direction as in Phase 1) to the speed defined in machine data: 

$$ \text{MD35400 $\text{MA\_SPIND\_OSCILL\_DES\_VELO}$} \ . $$

$\text{Time } t_1$ ($t_2$) starts. The process continues with Phase 2.

---

**Oscillation via PLC**

NC/PLC IS:

$$ \text{DB31, ... DBX18.4 (oscillation via PLC)} \ and $$

$$ \text{DB31, ... DBX18.5 (oscillation speed)} \ $$

accelerates the spindle motor to the speed (with oscillation acceleration) defined in machine data:

$$ \text{MD35400 $\text{MA\_SPIND\_OSCILL\_DES\_VELO}$ (oscillation speed)} \ . $$

The direction of rotation is defined by NC/PLC IS:

$$ \text{DB31, ... DBX18.7 (set direction of rotation CCW)} \ and $$

$$ \text{DB31, ... DBX18.6 (set direction of rotation CW)} \ . $$

The oscillation movement and the two times $t_1$ and $t_2$ (for clockwise and counter-clockwise rotation) must be simulated on the PLC.

---

**Special features**

Setting/Resetting the NC/PLC IS and machine data in oscillation mode:

- To decelerate the spindle, the PLC user need not set NC/PLC IS:
  $$ \text{DB31, ... DBX4.3 (spindle stop)} \ . $$
  The spindle is brought to a standstill internally by the control when a gear stage change is requested.

- The gear stage change should always be terminated with NC/PLC IS:
  $$ \text{DB31, ... DBX16.3 (gear changed)} \ . $$

- NC/PLC IS:
  $$ \text{DB31, ... DBX18.5 (oscillation speed)} \ $$
  should be used to support mechanical engagement of the gear.
  It has no effect on the internal control mechanism for the gear stage change procedure and should therefore only be set as necessary.
End of oscillation mode

On termination of oscillation mode, the spindle returns to open-loop control mode and automatically changes to the mode defined by $SPCON$ or $SPCOF$.

All gear-specific limit values (min./max. speed, etc.) correspond to the set values of the actual gear stage.

Functionality

Machine tools of conventional design require a gear stage of the spindle in oscillation mode.

If the machine data configuration is:

$MD35010 \$MA\_GEAR\_STEP\_CHANGE\_ENABLE = 1$

the following sequence is implemented:

- Deceleration of the spindle.
  The braking action corresponds to an M5 movement.

- Output of VDI interface signals:
  DB31, ... DBX84.6 (oscillation mode)
  DB31, ... DBX82.3 (change gear)
  DB31, ... DBX82.0-82.2 (set gear stage A to C).
  If position control has been enabled, it is disabled:
  DB31, ... DBX61.5 = 0.

- The load gearbox can now "disengage".

- NC/PLC IS:
  DB31, ... DBX18.5 (oscillation enable)
  can be set by the PLC.
  The spindle motor then performs an oscillation motion with preset values.
  The oscillation motion is designed to facilitate and accelerate the re-engaging of the gear wheels.
• Writing of NC/PLC IS:
  DB31, ... DBX16.0-16.2 (actual gear stage A to C)
  by the PLC.

• Once the PLC has sent:
  DB31, ... DBX16.3 (gear changed)
  to the NCK, the last movement to be active is continued, if available.
  For indirect encoders (motor encoders), the homing status is cleared:
  DB31, ... DBX60.4/5 = 0.

### Block change

If the spindle is switched to oscillation mode
with NC/PLC IS:
DB31, ... DBX82.3 (change gear),
the processing of the part program remains suspended.
A new block is not executed.

If oscillation mode is terminated with NC/PLC IS:
DB31, ... DBX16.3 (gear changed),
the processing of the part program is resumed.
A new block is executed.

![Figure 16-5 Block change following oscillation mode](image-url)
Oscillation mode

Typical time sequence for the gear stage change with a spindle:

$t_1$: With the programming of S1300, NCK detects a new gear stage (second gear stage), sets NST DB31, ..., DBX82.3 (change gear) and blocks processing for the next part program block (= internal feed disable*).

$t_2$: The spindle is stationary, and oscillation starts (oscillation via the NCK). NST DB31, ..., DBX18.5 (oscillation speed) must be set at the latest by $t_2$.

$t_3$: The new gear stage is engaged. The PLC user transfers the new (actual) gear stage to the NCK and sets NST DB31, ..., DBX16.3 (gear is changed).
tₚ: NCK then retracts NST DB31, ... DBX82.3 (change gear), ends the oscillation, releases the next part program block for processing, and accelerates the spindle to the new S value (S1300).

*: The internal feed disable is set if:
  - The spindle gear stage change has been programmed via the part program and
  - A processing block is activated (i.e. G0 is not active)

The internal feed disable is not set during a gear stage change from synchronized actions or in the case of specifications via the PLC with FC 18.

Figure 16-6  Gear stage change with stationary spindle

16.4.8  Gear stage change at fixed position

Application and advantages

Machine tools increasingly use standardized spindle drives, firstly to save technological dead time on a gear stage change and secondly to gain the cost benefits of using standardized components.

The "Gear stage change at fixed position" function supports the "directed gear stage change" of load gearboxes that need to be activated in a different way than the NC. The gear stage change can in this case only be performed at a defined spindle position. An oscillation movement as required by conventional load gearboxes is thus no longer necessary.

Sequence for gear stage change at fixed position

The gear stage change at fixed position
Machine data configuration:

\texttt{MD35010 \$MA\_GEAR\_STEP\_CHANGE\_ENABLE = 2}

runs the following sequence:

- Positioning of the spindle from standstill or movement to the position configured in machine data:
  \texttt{MD35012 \$MA\_GEAR\_STEP\_CHANGE\_POSITION}.
  If the gear stage change is performed out of a movement, then the current direction of rotation is maintained. The spindle is in positioning mode during the positioning action.
  NC/PLC IS:
  \texttt{DB31, ... DBX84.5 (positioning mode)}
  is output.
  If no reference is available:
  \texttt{DB31, ... DBX60.4/5 = 0}
  or NC/PLC IS:
  \texttt{DB31, ... DBX17.4/5 (resynchronize on positioning MS 1/2)}
  is set, the positioning action is extended by the time it takes to find the zero mark.

- After reaching the gear stage change position configured in machine data:
  \texttt{MD35012\$MA\_GEAR\_STEP\_CHANGE\_POSITION}
  the machine waits for the time in machine data:
  \texttt{MD35310 \$MA\_SPIND\_POSIT\_DELAY\_TIME}
  before switching to oscillation mode,
  and the known gear stage change dialog starts.

- Output of NC/PLC interface signals:
  \texttt{DB31, ... DBX84.6 (oscillation mode)}
  \texttt{DB31, ... DBX82.3 (change gear)}
  \texttt{DB31, ... DBX82.0-82.2 (set gear stage A to C)}.

- Position control is not disabled when an active measuring system with indirect encoder (motor encoder) is used:
  \texttt{MD31040 \$MA\_ENC\_IS\_DIRECT = 0}
  If a measuring system with a direct encoder (load encoder) is active, position control is deactivated:
  \texttt{DB31, ... DBX61.5 = 0},
  because the induction flux to the load is interrupted and closed-loop position control is no longer possible.

- If position-controlled operation is not possible, it can be disabled by resetting "Controller enable":
  \texttt{DB31, ... DBX2.1 = 0}

- Mechanical switchover of the gear stage on the machine.
  No oscillation movement is required from the drive.
  NC/PLC IS:
  \texttt{DB31, ... DBX18.5 (oscillation enable)}
  and
  \texttt{DB31, ... DBX18.4 (oscillation via PLC)}
  should not be set.
  In principle, oscillation movement is still possible at this point.
• Writing of NC/PLC IS:
  DB31, ... DBX16.0-16.2 (actual gear stage A to C)
  by the PLC.

• After signal:
  DB31, ... DBX16.3 (gear stage changed),
  the last movement to be active is continued, if available.
  For indirect encoders (motor encoders), the referencing status is cleared:
  DB31, ... DBX60.4/5 = 0.
  The spindle is in speed control mode and NC/PLC IS:
  DB31, ... DBX84.7 (open-loop control mode)
  is output.
GSC at fixed position

Typical time sequence for the gear stage change at fixed position:

- **t₁**: With the programming of S1300, NCK detects a new gear stage (second gear stage), NCK sets NST DB31, ... DBX84.5 (positioning mode) and blocks processing for the next part program block (= internal feed disable*).
- **t₂**: The spindle is stationary, and exact stop is signaled.
- **t₃**: Gear stage change - wait time
t₄: The new gear stage is engaged. The PLC user transfers the new (actual) gear stage to the NCK and sets NST DB31, ... DBX16.3 (gear is changed).

t₅: NCK then retracts NST DB31, ... DBX82.3 (change gear), releases the next part program block for processing, and accelerates the spindle to the new S value (S1300).

* : The internal feed disable is set if:

- The spindle gear stage change has been programmed via the part program and
- A processing block is activated (i.e. G0 is not active)

The internal feed disable is not set during a gear stage change from synchronized actions or in the case of specifications via the PLC with FC 18.

Figure 16-7 Gear stage change with stationary spindle

gear stage change position MD35012

The gear stage change position is defined in machine data:

MD35012 $MA_GEAR_STEP_CHANGE_POSITION

for each gear stage.

Gear stage change wait time MD35310

After the positioning action the machine waits for the time configured in machine data:

MD35310 $MA_SPIND_POSIT_DELAY_TIME

until gear change request:

DB31, ... DBX84.6 (oscillation mode)
DB31, ... DBX82.3 (change gear)

and

DB31, ... DBX82.0-82.2 (set gear stage A to C)

are output.

Position identifiers / position

The position is always approached via the shortest path (corresponds to DC).

If no reference is available and the spindle is in standstill (e.g. after Power On), then the direction of travel is determined by the following machine data:

MD35350 $MA_SPIND_POSITIONING_DIR

If an adjustable gear stage change position is required, then this can be achieved by writing the machine data and by a subsequent NewConfig.

The change of the MD value can be achieved by the part program or HMI.

If the system is unable to reach the preset position, then alarm 22020 is signaled and the gear stage change dialog between NCK and PLC does not take place in order not to destroy the gears. As this alarm is serious, the part program cannot continue and the cause must be eliminated under all circumstances. Experience has shown that the abortion of positioning is usually due to incorrect MD settings or incompatible PLC signals.

Speed

The positioning speed is taken from the machine data which is configured depending on the gear stage:
MD35300 $MA_SPIND_POSCTRL_VELO

NC/PLC IS "Spindle speed override"/"Feedrate override" at DB31, ... DBX17.0=0: DB31, ... DBB19
as well as:
DB31, ... DBX17.0=1: DB31, ... DBB0
are effective as normal for positioning.
The positioning speed can be changed proportionally through the program statement OVRA[Sn].

**Note**

OVRA[Sn] is valid modally. After the gear stage change, a value appropriate for the machining should be reset.

The part program statement FA[Sn] does not change the positioning speed during gear stage change.

**Acceleration**

The acceleration values are determined by the machine data which is configured depending on the gear stage:

MD35200 $MA_GEAR_STEP_SPEEDCTRL_ACCEL
and
MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL

The acceleration can be changed proportionally by programming ACC[Sn].

**Note**

ACC[Sn] is valid modally. After the gear stage change, a value appropriate for the machining should be reset.

**Speed-dependent acceleration**

The "knee-shaped acceleration characteristic" is effective as in positioning with SPOS or FC18.

**Jerk**

It is currently not possible to limit the change in acceleration.

**End of positioning**

The transition between the end of the positioning action (DB31, ... DBX84.5) and the start of oscillation mode (DB31, ... DBX84.6) is defined on reaching "Exact stop fine" (DB31, ... DB60.7) and the time value entered in machine data: MD3510 $MA_SPIND_POSIT_DELAY_TIME.

S1: Spindles

16.4 Configurable gear adaptation
The determination of the transition condition has an effect firstly on the gear stage change time and secondly on the accuracy of the approach to the preset gear stage change position.

Block change

The block change is stopped and the machining blocks are not started until the gear stage has been changed by the PLC (DB31, ... DBX16.3).

End of gear stage change

Once the gear stage change has been completed, the spindle returns to open-loop control mode and will automatically change to the closed-loop control mode defined by SCON or SPCOF.

All gear-specific limit values (min./max. speed of gear stage, etc.) correspond to the check-back values of the actual gear stage.

Supplementary conditions

- The spindle must have at least one measuring system.
- Position-controlled operation must be possible and must have been activated.
- Generally, it must be possible to execute SPOS from the part program, from a synchronized action or via FC18: “Start spindle positioning" without errors.

Unless all requirements can be met, the function described cannot be used successfully.

Activation

The function of gear stage change at fixed position is activated by the configuration:

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 2

16.4.9 Configurable gear step in M70

Technical background

In some machines the spindle needs to be in a particular gear stage during axis mode.

Possible reasons:

- Only one optimization (Kv, feedforward control, filter) to suit a gear stage can be found in the servo parameter set for axis mode (index 0). The machine data for this parameter set should not be rewritten.

- There is only one mechanical transmission ratio which, unlike the others, possesses little or no backlash compensation. The spindle can only follow a path motion or transformations (e.g. TRANSMIT) together with other axes in this gear step.
**Function**

If the function is activated, a predefined gear step is loaded automatically during transition to axis mode.

The gear step change is integrated into the M70 process and occurs after spindle deceleration and before the loading of the servo parameter set with index 0 (note MD35590 $MA_PARAMSET_CHANGE_ENABLE!).

The typical dialog between NC and PLC which occurs during gear step changes is executed in a similar way to programmed gear step changes (M41 ... M45) performed.

**Preconditions**

Gear step changes during transition to axis mode require general enabling of the gear step change via the machine data:

- MD35010 $MA_GEAR_STEP_CHANGE_ENABLE (assign parameters to the gear step change).
- MD35090 $MA_NUM_GEAR_STEPS (number of gear steps set up)

**Activation/deactivation**

The function is activated/deactivated via the machine data:

- MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE (gear step for axis mode in M70)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No implicit gear step change occurs in M70. The current gear step is retained (default setting!).</td>
</tr>
<tr>
<td>1 ... 5</td>
<td>A gear stage change to gear stage 1 ... 5 occurs during the processing of M70.</td>
</tr>
</tbody>
</table>

**Supplementary conditions**

- **Gear step change at fixed position** (MD35010 $MA_GEAR_STEP_CHANGE_ENABLE = 2)

  The "gear step change at fixed position" function is supported. The sequence in M70 is then extended by the time it takes to position the spindle. The position is approached at the current gear step.

- **Transition to axis mode without programming M70**

  The control system detects the transition to axis mode automatically from the program sequence (see "Implicit transition to axis mode (Page 1224)") and generates the requisite M70 sequence, including the gear step change, within the control system.

- **Transition to axis mode with FC 18**

  Implicit gear step change is not supported in transition to axis mode with the FC 18 ("Start axis"). This requires the right gear step to be engaged by the PLC application before switching to axis mode. The gear step change is also possible with the FC 18 ("Start gear step change").

- **Change from axis mode to spindle mode**

  When changing from axis mode to spindle mode, the gear step loaded with M70 remains activated. The gear stage which is activated in spindle mode prior to M70 is not automatically
loaded again. The servo parameter set is changed from parameter set 1 (index 0) to parameter sets 2 ... 6 (index 1 ... 5) to suit the gear step (with MD35590 $MA_PARAMSET_CHANGE_ENABLE < 2).

Example

Gear step 4 should be loaded in the case of spindle transition to axis mode.

Configuration: MD35014 $MA_GEAR_STEP_USED_IN_AXISMODE[<spindle name>] = 4

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N05 M3 S1000</td>
<td></td>
</tr>
<tr>
<td>N10 G1 X100 F1000</td>
<td></td>
</tr>
<tr>
<td>N15 M70 ; Gear step 4 is loaded.</td>
<td></td>
</tr>
<tr>
<td>N20 POS(C)=77</td>
<td></td>
</tr>
<tr>
<td>N25 ...</td>
<td></td>
</tr>
</tbody>
</table>

Note

MD35014 can be changed in the NewConfig. Thus, the gear step being loaded can still be changed in the part program before transition to axis mode, if necessary.

16.4.10 Suppression of the gear stage change for DryRun, program test and SERUPRO

Function

For test feed rate (DryRun), program test and SERUPRO, normally, a gear stage change is not required. This is the reason that it can be suppressed for these functions. The corresponding configuration is realized with bits 0 ... 2 in machine data:

MD35035 $MA_SPIND_FUNCTION_MASK

<table>
<thead>
<tr>
<th>Dry run feedrate (DryRun)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 = 0</td>
</tr>
<tr>
<td>Bit 0 = 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program test and SERUPRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1 = 0</td>
</tr>
</tbody>
</table>
Bit 1 = 1  For active program test / SERUPRO function - for part program blocks - a gear stage change is suppressed with M40, M41 to M45, programming via FC18 and synchronized actions.

| Bit 2 = 0 | Gear stage change for programmed gear stage is not performed subsequently on REPOS after deselection of functions DryRun, Program Test and SERUPRO. |
| Bit 2 = 1 | Gear stage change for programmed gear stage is performed after deselection of functions DryRun and SERUPRO if possible. |

**Sequence**

If a gear stage change is suppressed, if necessary, the interpolator will limit the programmed spindle speed to the permissible speed range of the active gear stage.

NC/PLC interface signals DB31, ... DBX83.2 (setpoint speed increased) and DB31, ... DBX83.1 (setpoint speed limited) generated as a result of this limit are suppressed.

Monitoring by the PLC program is not necessary during DryRun and in dry run feedrate.

When the gear stage change is suppressed, no new gear stage setpoint (DB31,... DBX82.0-82.2) is output to the PLC.

The gear stage change request DB31, ... DBX82.3 (change gear) is also suppressed.

This ensures that no gear stage change information has to be processed by the PLC program.

**Determining the last active gear stage**

System variable $P\_GEAR returns the gear stage programmed in the part program (which may not have been output to the PLC).

System variable $AC\_SGEAR can be used to read the last active gear stage from the part program, synchronized actions and at the user interface.

**Behavior after deselection**

The DryRun function can be deselected within a running part program. Once it has been deselected, the correct gear stage requested by the part program must be identified and selected.

It cannot be assured that the remainder of the part program will run without errors until the correct gear stage has been activated. Any necessary gear stage change is performed in the system REPOS started on deselection if the spindle is in speed control mode. A complete gear stage change dialog takes place with the PLC and the last programmed gear stage is activated.

If, for REPOS, there is a mismatch between the gear stage programmed in the part program and the actual gear stage supplied via the NC/PLC interface, then no gear stage change takes place.

The same applies to the SERUPRO function.

Further explanations regarding the block search function SERUPRO, see:

**References:**
Function Manual, Basic Functions; Mode Group, Channel, Program Mode, Reset Response (K1)

Boundary conditions

If the gear stage change is suppressed, the output spindle speed moves within the speed range specified by the current gear stage.

The following restrictions apply to the subsequent activation of a gear stage change with REPOS:

- The gear stage change is not activated subsequently if the spindle in the deselection or target block is a command spindle (synchronized action) or PLC spindle (FC18).
- If the gear stage cannot be activated because the spindle is in position or axis mode or a link is active, alarm 22011"Channel%1 block%3 spindle2% Change to programmed gear stage not possible" is signaled.

Example

Gear stage change in DryRun

```plaintext
; 1. Activate 1st gear stage for output state
N00 M3 S1000 M41 ; 1st gear stage is selected
M0 ; Part program stops

; PI service: Activate dry run feedrate (DryRun)
N10 M42 ; (Configuring)
N11 G0 X0 Y0 Z0 ; Positioning axes
N12 M0 ; Part program stops

; PI service: Deactivate dry run feedrate (DryRun)
N20 G1 Z100 F1000 ; REORG and REPOS are performed
; now the gear stage change to the 2nd gear stage takes place
...
N99 M30 ; Part program end
```

16.5 Additional adaptations to the spindle functionality that can be configured

Spindle-specific functions are set using machine data:

MD35035 $MA_SPIND_FUNCTION_MASK (spindle functions)
MD35035 is bit coded:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ... 2</td>
<td>Gear stage change behavior for test feedrate (DryRun), program test and SERUPRO. See &quot;Suppression of the gear stage change for DryRun, program test and SERUPRO (Page 1265)&quot;.</td>
</tr>
<tr>
<td>4</td>
<td>The programmed speed $s_{...}$ including speed setpoints FC18 and synchronized actions can be accepted in setting data SD43200 $SA_{SPIND_S}$ (speed for spindle start via PLC interface). See &quot;Special spindle motion via the PLC interface (Page 1277)&quot;.</td>
</tr>
<tr>
<td>5</td>
<td>For bit 5 = 1, the content of setting data SD43200 $SA_{SPIND_S}$ acts as the speed setpoint for JOG. You can use the JOG keys to operate the spindle at the speed defined in SD43200. If the content is zero, other JOG speed setpoints are active (see SD41200 $SN_JOG_SPIND_SET_VELO$).</td>
</tr>
<tr>
<td>8</td>
<td>The programmed cutting velocity $s_{...}$ including setpoints via FC18 and synchronized actions can be accepted in setting data SD43202 $SA_{SPIND_CONSTCUT_S}$ (cutting rate for spindle start via the PLC interface). See &quot;Special spindle motion via the PLC interface (Page 1277)&quot;.</td>
</tr>
<tr>
<td>10</td>
<td>For the master spindle, the value of the 15th G group (feedrate type) can be accepted in setting data SD43206 $SA_{SPIND_SPEED_TYPE}$ (spindle speed type for spindle start via the PLC interface). See &quot;Special spindle motion via the PLC interface (Page 1277)&quot;.</td>
</tr>
<tr>
<td>12</td>
<td>Bit 12 = 0: Spindle override is not effective for the zero mark search with M19, SPOS or SPOSA = 0. Bit 12 = 1: Spindle override is effective for the zero mark search with M19, SPOS or SPOSA = 0.</td>
</tr>
<tr>
<td>19</td>
<td>When programming SPOS and SPOSA, auxiliary function M19 can be implicitly generated and output to the PLC. See &quot;Positioning mode (Page 1205)&quot;.</td>
</tr>
<tr>
<td>20</td>
<td>In the case of M70 functionality which is generated in the control, auxiliary function M70 can be implicitly generated and output to the PLC. See &quot;Implicit transition to axis mode (Page 1224)&quot;.</td>
</tr>
</tbody>
</table>
| 22 | Bit 22 = 0: The NC/PLC interface signal DB31, ... DBX17.6 (invert M3/M4) also affects the function "interpolatory tapping (G331/G332)". **Note:** The following must be observed for this setting:  
- For part programs with G331 and G332, it is necessary to set NC/PLC interface signal DB31, ... DBX17.6 (invert M3/M4) **before part program start** to a stable value.  
- Existing application-based solutions, e.g. as are used in tapping cycles, must, if required, be adapted. It is known, for example, in tapping cycles, that the spindle direction of rotation can be inverted using MIRROR depending on a cyclic-specific setting data.  
  Bit 22 = 1: The NC/PLC interface signal DB31, ... DBX17.6 (invert M3/M4) does **not** affect the function "interpolatory tapping (G331/G332)". |

Changes to MD35035 become effective after an NC reset.
16.6 Selectable spindles

Function

The "selectable spindles" function allows you to write part programs with reference to the spindles used ("channel spindle, logical spindle") regardless of the actual assignment of configured spindles ("physical spindles") to a channel.

The physical spindles loaded or unloaded by "axis replacement" no longer have to be specified explicitly in the part program.

Spindle number converter (SD42800)

Each spindle is uniquely mapped to a machine axis using a configurable number:

\[
\text{MD35000} \, \text{MA_SPIND_ASSIGN_TO_MACHAX[AX...]} = <\text{spindle number}>
\]

This number always applies to one spindle, whereby it is of no importance in which channel the spindle is actively handled.

The channel spindles can be switched over because an intermediate level is introduced between the logical spindle numbers used in the part program and the physical spindles existing in the channel.

Every logical spindle used in the part program is assigned a physical spindle in a table comprising setting data:

\[
\text{SD42800} \, \text{SC_SPIND_ASSIGN_TAB[<n>]} \, (\text{spindle number converter}) = ...
\]

Index \(<n>\) corresponds to the programmed spindle number or the programmed address extension. The contents of the particular SD is the physical spindle that is actually available.

The spindle number converter is effective in spindle programming by means of:

- The part program
- Synchronized actions

The spindle number converter has no effect with PLC commands, which use function block FC18. The physical spindle must always be addressed there within the context of the axis.

Logical spindles can be changed over by changing SD42800. The changeover can be made from the part program, from the PLC and/or HMI.

Note

The logical master spindle is contained in setting data SD42800 $\text{SC_SPIND_ASSIGN_TAB[0]}$. It is only used for display purposes.

This setting data is defined in the part program by SETMS (logical spindle).

Unused spindles are assigned the value 0 in SD42800.

System variables, which are involved in the spindle changeover:

\[
\text{$P\_S, \, \text{P\_SDIR, \, P\_SMODE, \, P\_GWPS, \, AC\_SDIR, \, AC\_SMODE, \, AC\_MSNUM, \, AA\_S}}
\]

References:

Programming Manual, Job Planning
The converted, physical spindle number is always output as the address extension in the auxiliary function output.

**Supplementary conditions**

- Switchable channel spindles are **not** a substitute for the Axis replacement function.
- You can only switch spindles, which have been assigned to the channel by means of configuration.
- If spindles, which are presently active in another channel, are designated for switchover, either the "Auto-Get" function is triggered for the physical spindle or alarm 16105 "Assigned spindles do not exist" is output, depending on the configuration variant.
- If SD42800 $SC_SPIND_ASSIGN_TAB[<n>] is specified by the PLC or from HMI, then the channel whose table is changed should be in Reset status or the spindle to be changed should not be used in the running part program. A synchronized response can be achieved by means of a **STOPRE** preprocessor stop.
- The multiple mapping of logical to physical spindles is not prevented in the NC. However, with the display of logical spindle in the user interface, there are ambiguities corresponding to the change table.
- Spindle conversion operates on spindles via FC 18.

**Activation**

SD42800 $SC_SPIND_ASSIGN_TAB[<n>] is enabled by setting the following machine data setting:

```
MD20092 $MC_SPIND_ASSIGN_TAB_ENABLE=1
```

**Basic setting SD42800**

After switching on the NC with the commissioning switch in setting 1 (delete SRAM) SD42800 $SC_SPIND_ASSIGN_TAB[<n>] is in the basic setting.

The numbers of the logical and physical spindles are identical.

```
SD42800 $SC_SPIND_ASSIGN_TAB[1] = 1
SD42800 $SC_SPIND_ASSIGN_TAB[2] = 2
SD42800 $SC_SPIND_ASSIGN_TAB[3] = 3
SD42800 $SC_SPIND_ASSIGN_TAB[4] = 4
SD42800 $SC_SPIND_ASSIGN_TAB[5] = 5
... 
```

**Example**

Spindle configurations:

- Assignment, spindle number and machine axis:
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX [AX4] = 1
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX [AX5] = 2
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX [AX6] = 3
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX [AX7] = 5

• Accepting a machine axis in a channel:

MD20070 $MC_AXCONF_MACHAX_USED[0] = 4
MD20070 $MC_AXCONF_MACHAX_USED[1] = 5
MD20070 $MC_AXCONF_MACHAX_USED[2] = 6
MD20070 $MC_AXCONF_MACHAX_USED[3] = 7

• Specifying the master spindle:

MD20090 $MC_SPIND_DEF_MASTER_SPIND = 1

Spindle number converter:

MD20092 $MC_SPIND_ASSIGN_TAB_ENABLE=1 ; Activate spindle number converter
SD42800 $SC_SPIND_ASSIGN_TAB[0]=1 ; Master spindle as configured
SD42800 $SC_SPIND_ASSIGN_TAB[1]=1 ; Basic setting of the table
SD42800 $SC_SPIND_ASSIGN_TAB[2]=2
SD42800 $SC_SPIND_ASSIGN_TAB[3]=3
SD42800 $SC_SPIND_ASSIGN_TAB[4]=0 ; Logical spindle not assigned

M3 S1000 ; Address extension 1, M1-3 S1-1000 is output
The spindle with configured No. "1" (No. of the physical master spindle) rotates.
...
...
SD42800 $SC_SPIND_ASSIGN_TAB[1]=5 ; Assignment of logical spindle 1 to physical spindle 5
SD42800 $SC_SPIND_ASSIGN_TAB[2]=3 ; Assignment of logical spindle 2 to physical spindle 3.
Notice: physical spindle 3 has now been assigned twice. When programming logical spindles 2 and 3, physical spindle 3 is always addressed. In the basic machine displays, both spindles rotate.
SETMS(2) ; SD42800 $SC_SPIND_ASSIGN_TAB[0] = 2 defined internally by NCK.
SETMS:

SETMS(<n>): The spindle with the number <n> is the master spindle (may differ from the initial setting: MD20090 $MC_SPIND_DEF_MASTER_SPIND). The master spindle must be defined for the following functions:

- **G95:** Revolutionsal feedrate
- **G96 S.../G961 S...:** Constant cutting rate in m/min or feet/min
- **G97/G971:** Cancel G96/G961 and freeze last spindle speed
- **G63:** Tapping with compensating chuck
- **G33/G34/G35:** Thread cutting
- **G331/G332:** Tapping without compensating chuck
- **G4 S...:** Dwell time in spindle revolutions

The current master spindle setting can be retained via RESET / part program end and part program start. The setting is done via the machine data:

- **MD20110 $MC_RESET_MODE_MASK**
- **MD20112 $MC_START_MODE_MASK**
<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M3: M&lt;n&gt;=3:</td>
<td>Clockwise spindle rotation for the master spindle Clockwise spindle rotation for spindle number &lt;n&gt;</td>
</tr>
<tr>
<td>M4: M&lt;n&gt;=4:</td>
<td>Counter-clockwise spindle rotation for the master spindle Counter-clockwise spindle rotation for spindle number &lt;n&gt;</td>
</tr>
<tr>
<td>M5: M&lt;n&gt;=5:</td>
<td>Spindle stop without orientation for the master spindle Spindle stop without orientation for spindle number &lt;n&gt;</td>
</tr>
<tr>
<td>S...: S&lt;n&gt;=...:</td>
<td>Spindle speed in rpm for the master spindle Spindle speed in rpm for spindle number &lt;n&gt;</td>
</tr>
<tr>
<td>SVC=...: SVC[n]=...:</td>
<td>Cutting rate in m/min or feet/min for the master spindle Cutting rate in m/min or feet/min for the spindle &lt;n&gt;</td>
</tr>
<tr>
<td>SPOS=...: SPOS[n]=...:</td>
<td>Spindle positioning for the master spindle Spindle positioning for spindle number &lt;n&gt; The block change is only performed when the spindle is in position.</td>
</tr>
<tr>
<td>SPOSA=...: SPOSA[n]=...:</td>
<td>Spindle positioning for the master spindle Spindle positioning for spindle number &lt;n&gt; The block change is executed immediately. Spindle positioning continues, regardless of further part program processing, until the spindle has reached its position.</td>
</tr>
<tr>
<td>SPOS=DC(...): SPOS[n]=DC(...): SPOSA=DC(...): SPOSA[n]=DC(...):</td>
<td>The direction of motion is retained for positioning while in motion and the position approached. When positioning from standstill, the position is approached via the shortest path.</td>
</tr>
<tr>
<td>SPOS=ACN(...): SPOS[n]=ACN(...): SPOSA=ACN(...): SPOSA[n]=ACN(...):</td>
<td>The position is always approached with negative direction of motion. If necessary, the direction of motion is inverted prior to positioning.</td>
</tr>
<tr>
<td>SPOS=ACP(...): SPOS[n]=ACP(...): SPOSA=ACP(...): SPOSA[n]=ACP(...):</td>
<td>The position is always approached with positive direction of motion. If necessary, the direction of motion is inverted prior to positioning.</td>
</tr>
<tr>
<td>SPOS=IC(...): SPOS[n]=IC(...): SPOSA=IC(...): SPOSA[n]=IC(...):</td>
<td>The travel path is specified. The direction of travel is determined from the sign in front of the travel path. If the spindle is in motion, the direction of travel is inverted as necessary to allow traversing in the programmed direction. If the zero mark is crossed during traversing, the spindle is automatically synchronized with the zero mark if no reference is available or if a new one has been requested via an interface signal.</td>
</tr>
<tr>
<td>M19: M[n]=19:</td>
<td>Positioning the master spindle to the position in SD43240 Positioning spindle number &lt;n&gt; to the position in SD43240 The block change is only performed when the spindle is in position.</td>
</tr>
<tr>
<td>M70: M&lt;n&gt;=70:</td>
<td>Bring spindle to standstill and activate position control, select zero parameter set, activate axis mode for the master spindle for spindle number &lt;n&gt;</td>
</tr>
</tbody>
</table>
### S1: Spindles

#### 16.7 Programming

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SPCON:</strong></td>
<td>Spindle position control ON for the master spindle</td>
</tr>
<tr>
<td><strong>SPCON(&lt;n&gt;):</strong></td>
<td>for spindle number &lt;n&gt;</td>
</tr>
<tr>
<td><strong>SPCON(&lt;n&gt;,&lt;m&gt;):</strong></td>
<td>for spindle numbers &lt;n&gt; and &lt;m&gt;</td>
</tr>
<tr>
<td><strong>PCOF:</strong></td>
<td>Spindle position control OFF, activate speed control mode for the master spindle</td>
</tr>
<tr>
<td><strong>SPCOF(&lt;n&gt;):</strong></td>
<td>for spindle number &lt;n&gt;</td>
</tr>
<tr>
<td><strong>SPCOF(&lt;n&gt;,&lt;m&gt;):</strong></td>
<td>for spindle numbers &lt;n&gt; and &lt;m&gt;</td>
</tr>
<tr>
<td><strong>FPRAON(S&lt;n&gt;):</strong></td>
<td>Revolutionary feedrate for spindle &lt;n&gt; ON, derived from the master spindle</td>
</tr>
<tr>
<td><strong>FPRAON(S&lt;n&gt;,S&lt;m&gt;):</strong></td>
<td>The revolutionary feedrate value must be specified with FA[S&lt;n&gt;].</td>
</tr>
<tr>
<td><strong>FPRAOF(S&lt;n&gt;):</strong></td>
<td>Revolutionarly feedrate for spindle &lt;n&gt; OFF</td>
</tr>
<tr>
<td><strong>G25 S...:</strong></td>
<td>Programmable minimum spindle speed limitation for the master spindle</td>
</tr>
<tr>
<td><strong>G25 S&lt;n&gt;:</strong></td>
<td>for spindle number &lt;n&gt;</td>
</tr>
<tr>
<td><strong>G26 S...:</strong></td>
<td>Programmable maximum spindle speed limitation for the master spindle</td>
</tr>
<tr>
<td><strong>G26 S&lt;n&gt;:</strong></td>
<td>for spindle number &lt;n&gt;</td>
</tr>
<tr>
<td><strong>LIMS=...:</strong></td>
<td>Programmable maximum spindle speed limitation with G96, G961, G97 for the master spindle</td>
</tr>
<tr>
<td><strong>LIMS[&lt;n&gt;]=...:</strong></td>
<td>for spindle number &lt;n&gt;</td>
</tr>
<tr>
<td><strong>VELOLIM[&lt;spindle&gt;]=...:</strong></td>
<td>Programmable limiting of the configured gear stage dependent maximum speed</td>
</tr>
<tr>
<td></td>
<td>Using machine data (MD30455 $MA_MISC_FUNCTION_MASK, bit 6), when programming in the part program, it can be set as to whether VELOLIM is effective independent of whether used as spindle or axis (bit 6 = 1) - or is able to be programmed separately for each operating mode (bit 6 = 0). If they are to be separately effective, then the selection is made using the name when programming:</td>
</tr>
<tr>
<td></td>
<td>• Spindle name S&lt;n&gt; for spindle operating modes</td>
</tr>
<tr>
<td></td>
<td>• Axis name, e.g. &quot;C&quot;, for axis operation</td>
</tr>
<tr>
<td></td>
<td>The correction value refers to:</td>
</tr>
<tr>
<td></td>
<td>• Spindles in axis operation (if MD30455 Bit 6 = 0):</td>
</tr>
<tr>
<td></td>
<td>To the configured maximum axis velocity (MD32000 $MA_MAX_AX_VELO).</td>
</tr>
<tr>
<td></td>
<td>• Spindles in spindle or axis operation (if MD30455 bit 6 = 1):</td>
</tr>
<tr>
<td></td>
<td>To the maximum speed of the active gear unit stage (MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[&lt;n&gt;])</td>
</tr>
<tr>
<td></td>
<td>For further explanations about the programming of VELOLIM, see:</td>
</tr>
<tr>
<td></td>
<td>References:</td>
</tr>
<tr>
<td></td>
<td>Programming Manual, Work Preparation</td>
</tr>
<tr>
<td><strong>WAITS:</strong></td>
<td>Synchronization command for master spindle</td>
</tr>
<tr>
<td></td>
<td>The subsequent blocks are not processed until the spindle programmed in a previous NC block with SPOSA has reached its position (with exact stop fine).</td>
</tr>
<tr>
<td><strong>WAITS after M5:</strong></td>
<td>Wait until the spindle is stationary.</td>
</tr>
<tr>
<td><strong>WAITS after M3/M4:</strong></td>
<td>Wait until the spindle reaches its setpoint speed.</td>
</tr>
</tbody>
</table>
### Statement Description

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAITS(&lt;n&gt;):</td>
<td>Synchronization command for spindle number &lt;n&gt;</td>
</tr>
<tr>
<td>WAITS(&lt;n&gt;,&lt;m&gt;):</td>
<td>Synchronization command for spindle numbers &lt;n&gt; and &lt;m&gt;</td>
</tr>
<tr>
<td>FA[S&lt;n&gt;]:</td>
<td>Programming of positioning speed (axial feed) for spindle &lt;n&gt; in [deg/min]</td>
</tr>
<tr>
<td></td>
<td>With FA[S&lt;n&gt;]=0, the configured value takes effect once more: MD35300 $MA_SPIND_POSCTRL_VELO</td>
</tr>
<tr>
<td>OVRA[S&lt;n&gt;]:</td>
<td>Programming of the axial override value for spindle &lt;n&gt; in [%]</td>
</tr>
<tr>
<td>ACC[S&lt;n&gt;]:</td>
<td>Programming of the axial acceleration capacity for spindle &lt;n&gt; in [%]</td>
</tr>
<tr>
<td>SPI(&lt;n&gt;):</td>
<td>With SPI(&lt;n&gt;) a spindle number is converted into the data type AXIS according to machine data MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[]</td>
</tr>
<tr>
<td></td>
<td>SPI is used if axis functions are to be programmed with the spindle.</td>
</tr>
<tr>
<td></td>
<td>The following commands are possible with SPI:</td>
</tr>
<tr>
<td></td>
<td>• Frame commands:</td>
</tr>
<tr>
<td></td>
<td>− CTRANS()</td>
</tr>
<tr>
<td></td>
<td>− CFINE()</td>
</tr>
<tr>
<td></td>
<td>− CMIRROR()</td>
</tr>
<tr>
<td></td>
<td>− CSCALE()</td>
</tr>
<tr>
<td></td>
<td>• Velocity and acceleration values for following spindles:</td>
</tr>
<tr>
<td></td>
<td>− FA[SPI(&lt;n&gt;)]</td>
</tr>
<tr>
<td></td>
<td>− ACC[SPI(&lt;n&gt;)]</td>
</tr>
<tr>
<td></td>
<td>− OVRA[SPI(&lt;n&gt;)]</td>
</tr>
<tr>
<td></td>
<td>• System variables:</td>
</tr>
<tr>
<td></td>
<td>− $P_PFRAME[SPI(&lt;n&gt;),TR]=&lt;value&gt;</td>
</tr>
<tr>
<td></td>
<td>− $P_PFRAME=CTRANS(X,&lt;axis value&gt;,Y,&lt;axis value&gt;,SPI(&lt;n&gt;),&lt;axis value&gt;)</td>
</tr>
<tr>
<td></td>
<td>− $P_PFRAME=CSCALE(X,&lt;scale&gt;,Y,&lt;scale&gt;,SPI(&lt;n&gt;),&lt;scale&gt;)</td>
</tr>
<tr>
<td></td>
<td>− $P_PFRAME=CMIRROR(S&lt;n&gt;,Y,Z)</td>
</tr>
<tr>
<td></td>
<td>− $P_UBFR=CTRANS(A,10) : CFINE (19,0.1)</td>
</tr>
<tr>
<td></td>
<td>For further explanations about the programming of SPI, see:</td>
</tr>
<tr>
<td></td>
<td><strong>References:</strong> Programming Manual, Work Preparation.</td>
</tr>
</tbody>
</table>

| M40:               | Automatic gear stage selection for the master spindle                     |
| M<n>=40:           | Automatic gear stage selection for spindle number <n>                      |
| M41 to M45:        | Select gear stage 1 to 5 for the master spindle                            |
| M<n>=41 to M<n>=45:| Select gear stage 1 to 5 for spindle number <n>.                           |

**Note**

M functions M3, M4, M5, and M70 are not output in DB21, ... DBB194 and DBB202 if a spindle is configured in a channel. These M functions are offered as extended M functions in DB21, ... DBB68 ff. and in the relevant axis DBs, DB31, ... DBB86 ff.
More detailed explanations for programming the spindle can be found in:
- Programming Manual, Fundamentals

### 16.7.2 Programming via synchronized actions

M functions M40 to M45 can also be programmed in synchronized actions.

Please note:
- The programming of M40 ... M45 in the part program has no effect on the current status of the automatic gear step change of synchronized actions, and vice versa.
- When programming S values with M40, automatic gear step change is effective separately for synchronized actions and the part program.
- M40 is deactivated after POWER ON. The gear step is not adjusted if an S value is specified from a synchronized action.
- An M40 command programmed using synchronized actions always remains active for synchronized actions (modal) and is not reset on reset.
- M41 ... M45 selects first to fifth gear steps in accordance with the programming in the part program. An axis replacement is necessary in order to run the function. Once the gear step change has been performed, the spindle status is neutral (same response to M3, M4, M5 programming).

For further explanations regarding the programming of the spindle as well as spindle movements from synchronized actions, refer to:
- Programming Manual, Job Planning
- Function Manual, Synchronized Actions

### 16.7.3 Programming spindle controls via PLC with FC18 - only 840D sl

When the PLC specifies the direction of rotation and speed using FC18, the NCK can determine and select a gear step to match the speed. This is equivalent to the M40 functionality when programming via the part program.

The correct start code must be set when FC18 is called in a PLC user program, in order to activate gear step selection.

More detailed explanations regarding the programming of spindle controls by PLC with FC18 can be found in:
- Function Manual, Basic Functions, Basic PLC Program (P3)
16.7.4 Special spindle motion via the PLC interface

Note
The function is only available when using SINUMERIK Operate!

Why use a special spindle interface?
This function can be used to program the spindle via an axial PLC interface as an alternative to the FC18. The simplicity of the settings results in slightly restricted functionality. This functionality can be used preferably for simple control applications.

Functionality
Spindles can be started and stopped outside a part program that is being executed via the internal DBB30 spindle interface:

- DB31, ... DBX30.0 (spindle stop)
- DB31, ... DBX30.1 (spindle start, clockwise)
- DB31, ... DBX30.2 (spindle start, counter-clockwise)
- DB31, ... DBX30.3 (select gear stage)
  Supplementary signal to the signal "spindle start, clockwise/counter-clockwise"; determines the gear stage that matches the speed analog to M40 in the part program.
- DB31, ... DBX30.4 (spindle positioning start)

In order to start a spindle job, the channel handling the spindle must be in the acceptance state. A spindle job is always started on the Low-High edge of an internal DBB30 signal.

Generally, the internal DBB30 start signals do not have any meaning in the static state and do not prevent the spindle being programmed by FC18, synchronized actions, the part program or JOG traversing (e.g. when the STOP signal is statically at "1").

Requirements
In order that spindle jobs are accepted via the DBB30 spindle interface, the following conditions must be fulfilled (acceptance state):

- The channel state must be in the "interrupted" or "reset" mode:
  - DB21, ... DBX35.6 = 1 (channel state "interrupted")
  - DB21, ... DBX35.7 = 1 (channel state "reset")

- The program state must be in the "interrupted" or "canceled" mode:
  - DB21, ... DBX35.3 = 1 (program state "interrupted")
  - DB21, ... DBX35.4 = 1 (program state "interrupted")

These states will occur on RESET and in JOG mode.
At the start time, the spindle concerned must meet the following requirements:

- It must be in the state "Channel axis" or "Neutral axis" and must not be moved by means of the JOG keys.
- When the spindle is specified, no positioning may be carried out by FC18 or synchronized actions.

**Note**

Spindle job outside the acceptance range

Low-high edges outside the acceptance range will be ignored. No alarm message is output by the NCK. It can be conceivable that the acceptance range will be indicated to the operator by a user-side PLC application.

Spindle jobs outside the acceptance range can also be carried out using FC18 or ASUB.

**Multi-channel operation**

In the case of multi-channel operation, the spindle started by the PLC becomes active in the channel that handles the spindle at the appropriate moment.

The channel can be determined on the PLC side by reading the NC/PLC interface signals:

DB31, ... DBX68.0-68.3 (NC axis/spindle in channel A to D)

**Spindle definitions**

The spindle settings are retained after a change in mode (e.g. from JOG to AUTOMATIC). The spindle definitions are applied to the part program at the start of the program and can be modified again using part program operations.

Using the following settings in the machine data:

MD35035 $MA_SPIND_FUNCTION_MASK (spindle functions),
certain spindle definitions (speed or cutting speed, feedrate type) can be taken from the part program, synchronized action and FC18 and entered in the corresponding setting data:

| Bit 4 = 1 | The programmed speed $S... including speed setpoints via FC18 and synchronized actions are accepted in the following setting data:  
SD43200 $SA_SPIND_S (speed for spindle start via PLC interface)  
Programmed S values that are not programmed speed values are not accepted in the SD. These include, for example:  
• S value at constant cutting speed (G96, G961)  
• S value for rotation-related dwell time (G4) |
| --- | --- |
| Bit 8 = 1 | The programmed cutting speed $S... including setpoints via FC18 and synchronized actions are accepted in the following setting data:  
SD43202 $SA_SPIND_CONSTCUT_S (cutting speed for spindle start via PLC interface)  
Programmed S values that are not programmed cutting speed values are not accepted in the SD. These include, for example:  
• S value outside the constant cutting speed (G96, G961, G962)  
• S value for rotation-related dwell time (G4) |
| Bit 10 = 1 | For the master spindle, the value of the 15th G group (feedrate type) is accepted in the following setting data SD43206 $SA_SPIND_SPEED_TYPE.  
SD43206 $SA_SPIND_SPEED_TYPE (spindle speed type for spindle start via PLC interface)  
For all other spindles, the value in SD43206 remains unchanged. |

### Speed default

Speed defaults from part program, FC18 or synchronized actions are written to the following setting data from all the usual sources:

SD43200 $SA_SPIND_S (speed for spindle start via PLC interface)  
The setting data can be written to as follows:  
• Through speed programming  
• Through direct programming in the part program  
• Through HMI software

**Note**

The setting data is written immediately and asynchronously to part-program execution.
The following conditions apply when writing:

<table>
<thead>
<tr>
<th>Programming through:</th>
<th>Conditions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed programming</td>
<td>● MD35035 $MA_SPIND_FUNCTION_MASK Bit 4 = 1 must be set.</td>
</tr>
<tr>
<td></td>
<td>● Constant cutting speed G96, G961 must not be active.</td>
</tr>
<tr>
<td></td>
<td>● The following NC/PLC interface signal must be set: DB31, ... DBX 84.0 = 0 (constant cutting speed is active)</td>
</tr>
<tr>
<td>Direct programming in the part program</td>
<td>A programmed S-value and the value of the directly written SD can be outdated with respect to time. If this is the case, after programming the SD, the statement STOPRE should be executed.</td>
</tr>
<tr>
<td>HMI</td>
<td>Only positive values including zero can be accepted. Otherwise, a corresponding message is generated.</td>
</tr>
</tbody>
</table>

**Gear stage change and effect on speed**

In the current version, no gear stage change is triggered if the setpoint speed is out of the speed range of the gear stage (exceptions, see "M40: Automatic gear stage selection for speeds outside the configured switching thresholds (Page 1293)"). The usual speed limitations and the speed increase to the setpoint speed are active.

**Setpoint speed for JOG**

For the following MD configuration:
MD35035 $MA_SPIND_FUNCTION_MASK bit 5 = 1
the content from SD43200 $SA_SPIND_S is active as setpoint speed for JOG.

You can use the JOG keys to operate the spindle at the speed defined in SD43200.

If the content is zero, other JOG speed definitions are active (see SD41200 $SN_JOGSPIND_SET_VELO).

**Constant cutting speed setting**

Defaults for constant cutting speed from part program, FC18 or synchronized actions are written to the following setting data from all the usual sources:

SD43202 $SA_SPIND_CONSTCUT_S (cutting speed for spindle start via PLC interface)

**Requirements**

The requirement for the definition of a constant cutting speed to be active is:

● The spindle involved must be the master spindle in the channel that handles the spindle.

This condition is fulfilled if the following NC/PLC interface signal is set:

DB31, ... DBX84.0 = 1 (constant cutting speed is active)

**Writing from the part program**

When writing from the part program, the value for the constant cutting speed is interpreted as follows:

● If G710 is active in the 12th G group: Metric
● If G700 is set in the 12th G group: Inch as [feet/min]
For G70, G71 and when writing from an external (HMI), the setting in the machine data:
MD10240 $MN_SCALING_SYSTEM_IS_METRIC
determines the interpretation of the written values.

For further explanations regarding the measuring system (metric/inch) see Section "G2: Velocities, setpoint / actual value systems, closed-loop control (Page 317)".

**Definition via FC18**

If the constant cutting speed is defined via FC18, the setting of bit 6 in byte 2 in the "Signals to concurring positioning axes" area determines how the speed value (bytes 8 ... 11) is interpreted.

**Setting via synchronized actions**

For definition via synchronized actions, analog to the part program, the feedrate type defines how the S value is interpreted.

**Reading from part program and synchronized actions**

The programmed cutting speed value can be determined in the part program and in the synchronized actions by reading the following system variables:

- `$P_CONSTCUT_S[<n>]` (last programmed constant cutting speed)
- `$AC_CONSTCUT_S[<n>]` (actual constant cutting speed)

Defined range of values of the two new system variables: $RV = \{0, DBL_{Max}\}$

The programmed cutting speed value can also be read via the OPI interface.

**Definition of the spindle speed type for the master spindle**

Definition of the spindle speed type for the master spindle from part program, FC18 or synchronized actions are written to the following setting data from all the usual sources:

SD43206 $SA_SPIND_SPEED_TYPE (spindle speed type for spindle start via PLC interface)

The value range and the functionality correspond to the 15th G group (feedrate type).

Permissible values are G values: 93, 94, 95, 96, 961, 97 and 971.

Depending on the setting, for DB31, … DBX30.1/2 (spindle start, clockwise/counter-clockwise) either the speed from SD43200 $SA_SPIND_S or the cutting speed from SD43202 $SA_SPIND_CONSTCUT_S is active:

93, 94, 95, 97 and 971: The master spindle is started with the speed from SD43200.
96 and 961: The speed of the master spindle is obtained from the specified cutting velocity (SD43202) and the radius of the transverse axis.

**Definitions for spindle positioning**

The definitions for spindle positioning using DB31, … DBX30.4 (spindle positioning start) are read from the following setting data:

SD43240 $SA_M19_SPOS (spindle position for spindle positioning with M19)
SD43250 $SA_M19_SPOSMODE (spindle position approach mode for spindle positioning with M19)

16.7.5 External programming (PLC, HMI)

SD43300 and SD42600

The revolutional feedrate behaviour can be selected externally via the axial setting data:
SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE (Rotational feedrate for spindles)
in JOG operating mode using the channel-specific setting data
SD42600 $SC_JOG.Feed_PER_REV_SOURCE (Revolutional fedrate control in JOG mode).

The following settings can be made via the setting data:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;0</td>
<td>The machine axis number of the rotary axis/spindle from which the revolutional feedrate shall be derived.</td>
</tr>
<tr>
<td>-1</td>
<td>The revolutional feedrate is derived from the master spindle of the channel in which the axis/spindle is active in each case.</td>
</tr>
<tr>
<td>0</td>
<td>Function is deselected.</td>
</tr>
</tbody>
</table>

FPRAON (S2)

Revolutional feedrate for spindle S2 ON, derived from the master spindle

FPRAON (S2, A)

Revolutional feedrate for spindle S2 ON, derived from axis A.
The revolutional feedrate value must be specified with FA[Sn].

FPRAOF (S2)

Revolutional feedrate for spindle S2 OFF.

SPI(n)

It is also possible to program SPI(n) instead of SPI(Sn).
16.8 Spindle monitoring

16.8.1 Permissible speed ranges

The permissible speed range of a spindle results from the parameterized or programmed speed limit values and the active spindle function (G94, G95, G96, G961, G97, G971, G33, G34, G35, G331, G332, etc.).

![Figure 16-8 Ranges of spindle monitoring functions / speeds](image-url)
16.8.2 Axis/spindle stationary

Functions such as tool change, open machine doors, path feedrate enable, etc. are only possible at the machine when the spindle is stationary.

Function

The "axis/spindle stationary" state is reached if a setpoint is no longer generated and the spindle actual speed falls below the configured threshold value for "axis/spindle stationary":

\[ \text{MD36060 $MA\_STANDSTILL\_VELO\_TOL$ (max. velocity/speed for "axis/spindle stationary")} \]

If the spindle has come to a standstill, the following NC/PLC interface signal is set:

\[ \text{DB31, ... DBX61.4 (axis/spindle stationary)} \]

Effectiveness

Monitoring for spindle stop is effective in all spindle modes and in axis mode.

Deactivate path feed

If a spindle is stopped in the open-loop control mode (M5), then path feed is deactivated if the following machine data is set:

\[ \text{MD35510 $MA\_SPIND\_STOPPED\_AT\_IPO\_START$ (feedrate enable for spindle stopped)} \]

The path feed is re-enabled if the spindle comes to a standstill.

16.8.3 Spindle in setpoint range

Function

"Spindle in setpoint range" spindle monitoring checks whether:

- The programmed spindle speed is reached.
- The spindle is at a standstill:
  \[ \text{DB31, ... DBX61.4 (axis/spindle stationary) = 1} \]
- The spindle is still in the acceleration or deceleration phase.

In the spindle mode, open-loop control mode, the setpoint speed is compared with the actual speed. If the actual speed deviates by more than the spindle tolerance that can be entered via MD (refer below) then:

- The following axial NC/PLC interface signal is set to "0":
  \[ \text{DB31, ... DBX83.5 (spindle in setpoint range) = 0} \]
- The next machining block is not enabled (depending on the setting in \[ \text{MD35500 $MA\_SPIND\_ON\_SPEED\_AT\_IPO\_START$, see "Axis/spindle stationary (Page 1284)"} \].
Spindle setpoint speed

The spindle setpoint speed is derived from the programmed speed taking into account the spindle correction and the active limits.

If the programmed speed is limited or increased, this is displayed using DB31, ... DBX83.1 (setpoint speed limited) or DB31, ... DBX83.2 (setpoint speed increased) (see also "Minimum / maximum speed of the gear stage (Page 1285)"). The means that reaching the tolerance range of the setpoint speed is not prevented.

Tolerance range for setpoint speed

The tolerance range of the setpoint speed is defined by the spindle speed tolerance factor:
MD35150 $MA_SPIND_DES_VELO_TOL

Example:
MD35150 $MA_SPIND_DES_VELO_TOL = 0.1
⇒ The spindle actual speed may deviate ±10% from the setpoint speed.

The following NC/PLC interface signal is set to "1" if the spindle actual speed lies within the tolerance range:
DB31, ... DBX83.5 (spindle in setpoint range) = 1

Special case:
If the spindle speed tolerance is set to "0", then DB31, ... DBX83.5 (spindle in the setpoint range) is permanently set to "1" and no path control is performed.

Speed change

Path control only takes place at the start of the traverse block and only if a speed change has been programmed. If the speed tolerance range is exited, e.g. due to an overload, the path movement is not automatically brought to a standstill.

16.8.4 Minimum / maximum speed of the gear stage

Minimum speed

The minimum speed of the gear stage of a spindle is configured in the machine data:
MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT[<n>]

The speed setpoints, generated taking into account the override, do not fall below the minimum speed.

If an S value is programmed, which is less than the minimum speed, the setpoint speed is increased to the minimum speed and the following NC/PLC interface signal is set:
DB31, ... DBX83.2 (speed setpoint increased)
The minimum gear stage speed is effective only in speed mode and can only be undershot by:

- Spindle override 0%
- M5
- S0
- DB31, ... DBX4.3 (spindle stop)
- DB31, ... DBX2.1 (withdraw controller enable)
- DB21, ... DBX7.7 (reset)
- DB31, ... DBX2.2 (delete distance-to-go / spindle reset)
- DB31, ... DBX18.5 (oscillation speed)
- DB21, ... DBX7.4 (NC stop axes plus spindles)
- DB31, ... DBX1.3 (axis/spindle disable)
- DB31, ... DBX16.7 (delete S value)

**Maximum speed**

The maximum speed of the gear stage of a spindle is configured in machine data:

MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[<n>]

The speed setpoints, generated taking into account the override, are limited to this speed.

The following NC/PLC interface signal is set in the case that the speed is limited:

DB31, ... DBX83.1 (speed setpoint limited)

**16.8.5 Diagnosis of spindle speed limitation**

**Function**

The limit or increase of the spindle speed is signaled by the output of the following NC/PLC interface signals:

- DB31, ... DBX83.1 (setpoint speed limited)
- DB31, ... DBX83.2 (setpoint speed increased)

To diagnose the effective/limiting spindle parameters, one can have read access to the most important parameters of spindle motion via system variables. The system variables are indexed with the spindle number and they return only values that are relevant to the speed control and spindle position modes.
The following system variables are available in the spindle mode:

<table>
<thead>
<tr>
<th>System variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_{\text{SMAXVELO}}[&lt;n&gt;]</td>
<td>Maximum possible spindle speed [rpm] resulting from the active limiting data.</td>
</tr>
<tr>
<td>$AC_{\text{SMAXVELO}}_{\text{INFO}}[&lt;n&gt;]</td>
<td>Specification of the speed limiting data as numerical value.</td>
</tr>
<tr>
<td>$AC_{\text{SMINVELO}}[&lt;n&gt;]</td>
<td>Minimum possible spindle speed [rpm], corresponds to the minimum speed in the speed control mode.</td>
</tr>
<tr>
<td>$AC_{\text{SMINVELO}}_{\text{INFO}}[&lt;n&gt;]</td>
<td>Specification of the speed increasing data as numerical value.</td>
</tr>
<tr>
<td>$AC_{\text{SMAXACC}}[&lt;n&gt;]</td>
<td>Acceleration value of spindle [r/s$^2$].</td>
</tr>
<tr>
<td>$AC_{\text{SMAXACC}}_{\text{INFO}}[&lt;n&gt;]</td>
<td>Acceleration limiting cause in the form of a numerical value.</td>
</tr>
<tr>
<td>$AC_{\text{SPIND_STATE}}[&lt;n&gt;]</td>
<td>Status bits of the spindle.</td>
</tr>
</tbody>
</table>

<n>: Spindle no. (n= 0: the variables are related to the current master spindle)

* The text of the numerical value should be taken from from the description of the system variables.

Evaluation of diagnosis data:
The system variables for each spindle can be read via synchronized actions and in the part program, giving due consideration to the preprocessing stop in the NCK.

**Boundary conditions**

The values delivered by the system variables depend on the spindle mode:

- **Speed control mode:**
  All system variables deliver current values.

- **Positioning mode:**
  The system variables $AC_{\text{SMAXVELO}}, AC_{\text{SMAXACC}}$ and $AC_{\text{SPIND\_STATE}}$ deliver valid values. The system variables $AC_{\text{SMINVELO}}$ and $AC_{\text{SMINVELO}}_{\text{INFO}}$ deliver the data that becomes effective on changing to the speed control mode.

- **Axis mode (e.g. if the spindle is used by a transformation TRANSMIT, TRACYL,... or follows a path motion as a special axis):**
  The system variable $AC_{\text{SPIND\_STATE}}$ can also be used in the axis mode. Separate system variables are available in the axis mode for dynamic data: $\text{AA\_VMAXM}, \text{AA\_VMAXB}$ and $\text{AA\_VLFCT}$.

The following control response is obtained for a type SERUPRO block search:

- The system variable $AC_{\text{SMAXVELO}} / AC_{\text{SMAXACC}}$ delivers the maximum representable speed / acceleration.

- $AC_{\text{SMAXVELO}}_{\text{INFO}}$ and $AC_{\text{SMAXACC}}_{\text{INFO}}$ deliver the VALUE "0" (no limitation is active).

- $AC_{\text{SMINVELO}}$ and $AC_{\text{SMINVELO}}_{\text{INFO}}$ deliver data as in case of normal part program processing.

- $AC_{\text{SPIND\_STATE}}$ returns the states as they are set for SERUPRO.
Example

Example of the visualization of the content of the system variables for Spindle 1. The variables are written to the R parameters cyclically. These can be displayed on HMI in the R Parameters area.

<table>
<thead>
<tr>
<th>Program code</th>
</tr>
</thead>
<tbody>
<tr>
<td>N05 IDS=1 WHENEVER TRUE DO $R10=$AC_SMAXVELO[1]</td>
</tr>
<tr>
<td>N10 IDS=2 WHENEVER TRUE DO $R11=$AC_SMAXVELO_INFO[1]</td>
</tr>
<tr>
<td>N15 IDS=3 WHENEVER TRUE DO $R12=$AC_SMINVELO[1]</td>
</tr>
<tr>
<td>N20 IDS=4 WHENEVER TRUE DO $R13=$AC_SMINVELO_INFO[1]</td>
</tr>
<tr>
<td>N25 IDS=5 WHENEVER TRUE DO $R14=$AC_SPIND_STATE[1]</td>
</tr>
</tbody>
</table>

See also

Spindle in setpoint range (Page 1284)

16.8.6 Maximum spindle speed

Maximum spindle speed: parameterizable, machine-dependent limit value

The maximum machine-dependent spindle speed – for protecting the spindle chuck or the tool, for example – is parameterized using the following machine data:

MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed)

The spindle speed is monitored by the NC in the actual values, i.e. taking into account the current gear stage.

Note

Machine manufacturer

It is recommended that changes to the maximum spindle speed in machine data MD35100 only be made when the spindle is stationary. This is particularly important for spindles that are active after NC reset (see also MD35040 SPIND_ACTIVE_AFTER_RESET). Otherwise alarm 22100 may be output.

References

- Manufacturer documentation: List Manual, "Detailed Machine Data Description"
- User documentation: Diagnostics Manual

Responses to violation

If the actual speed of a spindle exceeds the parameterized maximum spindle speed by more than the tolerance value,

"Actual spindle speed" > MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed) + MD35150 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance value)
the following effects can be seen on the NC side.

- DB31, ... DBX83.0 = 1 (speed limit exceeded)
- Alarm 22100 "Chuck speed exceeded" is displayed
- All axes and spindles of the channel are stopped

**Note**

**Rotary axis / spindle**

If a spindle is also temporarily operated as a rotary axis, alarm 22100 "Chuck speed exceeded" is displayed if:

"Actual speed of the rotary axis" > MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed) + MD35150 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance value)

Remedy: Adjust the maximum spindle speed to the maximum rotary axis speed during rotary operation:

MD35100 $MA_SPIND_VELO_LIMIT = MD32000 $MA_MAX_AX_VELO (maximum speed of the rotary axis)

For the change to take effect a reset must be triggered.

**Maximum spindle speed: Configurable process-dependent limit value**

The maximum process-related spindle speed is configured through the following setting data:

SD43235 $SA_SPIND_USER_VELO_LIMIT (maximum spindle speed)

Changes can be made by writing the setting data through:

- Programming in an NC program
- Manually through the user interface by the machine operator

Changes become effective immediately.

**Limitation of the speed setpoint**

Where appropriate, the controller limits programmed spindle speed setpoints to the values specified in the setting data. The limitation is then displayed through the following system variables:

$AC_SMAXVELO_INFO[<Spindle number>] == 21
16.8.7 Maximum encoder limit frequency

<table>
<thead>
<tr>
<th>CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limit violation</td>
</tr>
</tbody>
</table>

The maximum encoder frequency limit of the actual spindle position encoder is monitored by the control (the limit can be exceeded). It is the responsibility of the machine tool manufacturer to ensure that the configuration of the spindle motor, gearbox, measuring gearbox, encoder and machine data prevents the maximum speed of the actual spindle position encoder being exceeded.

Maximum encoder frequency exceeded.

If the spindle speed reaches a speed (large S value programmed), which exceeds the maximum encoder limit frequency (the maximum mechanical speed limit of the encoder must not be exceeded), the synchronization is lost. The spindle continues to rotate, but with reduced functionality.

With the following functions, the spindle speed is reduced until the active measurement system is operating below the encoder limit frequency again:

- Thread cutting (G33, G34, G35)
- Tapping without compensating chuck (G331, G332)
- Revolutional feedrate (G95)
- Constant cutting rate (G96, G961, G97, G971)
- SPCON (position-controlled spindle operation)

When the encoder limit frequency is exceeded

NC/PLC IS:
DB31, ... DBX60.4 (referenced/synchronized 1)
or
DB31, ... DBX60.5 (referenced/synchronized 2)

are reset for the measurement system in question and NC/PLC IS:
DB31, ... DBX60.2 (encoder limit frequency 1 exceeded)
or
DB31, ... DBX60.3 (encoder limit frequency 2 exceeded)

are set.

If the spindle is in axis mode, the maximum encoder limit frequency must not be exceeded. The maximum velocity (MD32000 $MA_MAX_AX_VELO) must lie below the maximum encoder limit frequency; otherwise, alarm 21610 is output and the axis is brought to a standstill.
### Maximum encoder limit frequency undershot

If the maximum encoder frequency limit has been exceeded and the speed subsequently falls below the maximum encoder limit frequency (smaller S value programmed, spindle offset switch changed, etc.), the spindle is automatically synchronized with the next zero mark or the next Bero signal. The new synchronization will always be carried out for the active position measuring system that has lost its synchronization and whose maximum encoder limit frequency is currently undershot.

### Special features

If the following functions are active, the maximum encoder frequency cannot be exceeded:

- Spindle positioning mode, axis mode
- Thread cutting (G33, G34, G35)
- Tapping without compensating chuck G331, G332 (does not apply to G63)
- Revolitional feedrate (G95)
- Constant cutting rate (G96, G961, G97, G971)
- SPCON

### 16.8.8 End point monitoring

**End point monitoring**

During positioning (the spindle is in positioning mode), the system monitors the distance from the spindle (with reference to the actual position) to the programmed spindle position setpoint (end point).

For this to work, in machine data:
- MD36000 $MA_STOP_LIMIT_COARSE (Exact stop limit coarse)
- MD36010 $MA_STOP_LIMIT_FINE (Exact stop limit fine)

Two limit values can be defined as an incremental path starting from the spindle position setpoint.

Regardless of the two limit values, the positioning of the spindle is always as accurate as defined by the connected spindle measurement encoder, the backlash, the transmission ratio, etc.
**Exact stop window dependent on parameter set**

Various parameter-set-dependent exact stop windows can be configured. This makes it possible to work to different levels of accuracy in axis mode and spindle positioning. The exact stop window can be configured separately for each gear step for spindle positioning.

![Exact stop zones of a spindle](image)

**DB31, ... DBX60.7 and DB31, ... DBX60.6 (position reached with exact stop coarse / fine)**

The two limit values defined by machine data:
- MD36000 $MA\_STOP\_LIMIT\_COARSE$ (Exact stop limit coarse)
- MD36010 $MA\_STOP\_LIMIT\_FINE$ (Exact stop limit fine)
are output to the PLC using NC/PLC IS:
- DB31, ... DBX60.7 (Position reached with exact stop coarse)
- DB31, ... DBX60.6 (Position reached with exact stop fine).

**Block change for SPOS and M19**

When positioning the spindle with SPOS or M19 the block is changed dependent on end point monitoring with NC/PLC IS:
- DB31, ... DBX60.6 (Position reached with exact stop fine).

All other functions programmed in the block must have achieved their end criterion (e.g., all auxiliary functions acknowledged by the PLC).

With SPOSA, the block change does not depend on the monitoring of the end point.
16.8.9 M40: Automatic gear stage selection for speeds outside the configured switching thresholds

Function

When M40 is active, an automatic gear stage selection is also made if the programmed spindle speed $S...$ lies outside the configured switching thresholds.

In this case, a distinction is made between the following cases:

- **Programmed speed too high**
  The programmed speed is higher than the configured maximum speed of the numerically largest gear stage:
  $$S... > MD35110 \ $MA\_GEAR\_STEP\_MAX\_VELO[n]$$
  In this case, the **highest gear stage** is selected (according to MD35090 $MA\_NUM\_GEAR\_STEPS$).

- **Programmed speed too low**
  The programmed spindle speed is less than the configured minimum speed of the first gear stage:
  $$S... < MD35120 \ $MA\_GEAR\_STEP\_MIN\_VELO[1]$$
  In this case, the **first gear stage** is selected.

- **Programmed speed = 0**
  When programming speed 0 ($S0$) the behavior depends on the configuration of the minimum speed of the first gear stage $MD35120 \ $MA\_GEAR\_STEP\_MIN\_VELO[1]$:
  - If $MD35120 \ $MA\_GEAR\_STEP\_MIN\_VELO[1] = 0$ is configured, then when programming $S0$, the **first gear stage** is selected.
  - If $MD35120 \ $MA\_GEAR\_STEP\_MIN\_VELO[1] > 0$ is configured, when programming $S0$ **no gear stage change** is performed and the last gear stage remains active. This means that it remains possible to stop the spindle with $S0$ (instead of $M5$) without initiating a gear stage change.

Effectiveness

Selecting the highest gear stage or the first gear stage for automatic gear stage selection (M40) is active when programming spindle speeds $S...$ using the part program, in synchronized actions or when entering via PLC FC18.

For speed programming from the part program for tapping with G331, the behavior is also supported for the second data set to select the gear stage (precondition: $MD35010 \ $MA\_GEAR\_STEP\_CHANGE\_ENABLE$, bit 5 = 1).

Boundary conditions

**Enabling the gear stage change**

The precondition for the function is that the gear stage change is generally enabled via machine data:

$MD35010 \ $MA\_GEAR\_STEP\_CHANGE\_ENABLE$ (assign parameters to the gear stage change)
MD35090 $MA_NUM_GEAR_STEPS (number of gear stages set up)

Example

Automatic gear stage selection M40 is the initial setting after NC reset.

Part program:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N15 S3500 M3</td>
<td>; S3500 is greater than MD35110 of the 2nd gear stage. The 2nd gear stage is selected.</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N50 S0 M3</td>
<td>; Spindle is stopped, S0 does not request a gear stage change (special handling, S0).</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N95 S5 M3</td>
<td>; S5 is less than MD35120 of the 1st gear stage. The 1st gear stage is selected.</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Configuring data for spindle 1 (AX5):

MD35010 $MA_GEAR_STEP_CHANGE_ENABLE[AX5] = 1 ; Enable gear stage change
MD35090 $MA_NUM_GEAR_STEPS[AX5] = 2 ; Number of existing gear stages
MD35110 $MA_GEAR_STEP_MAX_VELO[1,AX5] = 500 ; Upper switching threshold for gear stage 1
MD35120 $MA_GEAR_STEP_MIN_VELO[1,AX5] = 10 ; Lower switching threshold for gear stage 1
MD35110 $MA_GEAR_STEP_MAX_VELO[2,AX5] = 2000 ; Upper switching threshold for gear stage 2
MD35120 $MA_GEAR_STEP_MIN_VELO[2,AX5] = 500 ; Lower switching threshold for gear stage 2

16.9 Spindle with SMI 24 (Weiss spindle)

16.9.1 General Information

In order to be able to process the sensor data of the spindle in the control, the sensors must first be connected to I/O modules and transferred to the PLC via fieldbus (PROFIBUS DP or PROFINET I/O).

For a spindle with SMI 24 (Weiss spindle), the sensor data are transferred to the drive using DRIVE-CLiQ and are available there in drive parameters. When using cyclic drive telegram
139, sensor data from the drive are transferred to the control. They are then available there in the following system data:

- System variable
- OPI variables
- NC/PLC interface signals

**Requirement**

- The spindle is connected to the drive via Sensor Module SMI 24 using DRIVE-CLiQ.
- Drive telegram 139 is configured for the spindle.

---

**Note**

**Drive telegram 139**

In principle, a spindle with Sensor Module SMI 24 can also be operated with another drive telegram. However, sensor data is only transferred to the control using drive telegram 139.

---

### 16.9.2 Sensor data

#### Sensors in the spindle motor

The spindle sensors provide information about the clamping device and the angular position of the motor shaft:

- Analog sensor S1: Clamped state
  Voltage value in the range from 0 - 10 V depending on the position of the draw bar.

- Digital sensor S4: Piston end position
  - 0 = piston not in position
  - 1 = piston is in position, i.e. piston is free to move

- Digital sensor S5: Angular position of the motor shaft
  - 0 = motor shaft not aligned
  - 1 = motor shaft is in position (requirement: The spindle is stationary)

---

**Note**

**Spindle with sensor module SMI 24 and axis container**

A spindle with sensor module SMI 24 and drive telegram 139 for the transmission of sensor data to the control must not be part of an axis container whose axes are distributed over several NCUs via an NCU link.
Transmission of sensor data

Sensor data is transferred to the control from sensor module SMI 24 in cyclic drive telegram 139 as process data 11 - 14. Drive telegram 139 is based on drive telegram 136, where sensor data is transferred instead of the data of the 2nd encoder. A detailed description of drive telegram 139 can be found in:

References:
SINAMICS S120/S150 List Manual; Function Diagrams, Section: PROFIdrive

System data: Sensor data

Sensor data can be read into the control via the following system data:

<table>
<thead>
<tr>
<th>Meaning</th>
<th>System variable</th>
<th>NC/PLC interface</th>
<th>OPI variables</th>
<th>Drive parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor configuration</td>
<td>MOT_SENSOR_CONF[&lt;axis&gt;]</td>
<td>DBB132</td>
<td>vaMotSensorConf</td>
<td>r5000</td>
</tr>
<tr>
<td>Clamped state 1)</td>
<td>MOT_CLAMPING_STATE[&lt;axis&gt;]</td>
<td>DBW134</td>
<td>vaMotClampingState</td>
<td>r5001</td>
</tr>
<tr>
<td>Measured value analog sensor S1 2)</td>
<td>MOT_SENSOR_ANA[&lt;axis&gt;]</td>
<td>DBW136</td>
<td>vaMotSensorAna</td>
<td>r5002</td>
</tr>
<tr>
<td>Status digital sensors</td>
<td>MOT_SENSOR_DIGI[&lt;axis&gt;]</td>
<td>DBW138</td>
<td>vaMotSensorDigi</td>
<td>r5003</td>
</tr>
<tr>
<td>&lt;axis&gt;:</td>
<td>Machine axis name: AX1 ... AXn or spindle name: S1 ... Sm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) See Section "Clamped state (Page 1297)"
2) Sensor S1: 0 - 10 V
   Analog actual value: 0 - 10000 increments, resolution 1 mV
   Example:
   SIMATIC S7 input module: 0 - 27648 increments, resolution 0.36 mV
   Adaptation factor if you change to a spindle with SMI 24: 2,7648

Detailed system data description

For information about NC/PLC interfaces, see Section "Signals from axis/spindle (DB31, ...)
(Page 1642)".

References:
System variable: List Manual, System Variables
OPI variables: List Manual 2; Variables
Drive parameters: SINAMICS S120/S150 List Manual
16.9.3 Clamped state

Sensor S1 supplies an analog voltage value 0 V - 10 V depending on the position of the clamping device. The voltage value is available in the system data for evaluation of the clamped state on the user side.

**Note**

The subsequently described evaluation of sensor S1 to generate the state value for the clamped state and limiting the spindle speed are only realized if the following state values are displayed in drive parameter r5000:

- r5000.0 == 1: Sensors available
- r5000.1 == 1: Sensor S1 (clamped state) available
- r5000.10 == 1: State values are generated, speed limits p5043 active

See also Section "Sensor data (Page 1295)", paragraph: "System data: sensor data"

State value

To simplify the evaluation, the clamped state in the system data is also available as state value 0 - 11.

A voltage range corresponds to a certain clamped state. The voltage ranges can be set using drive parameter p5041[0...5].

A voltage tolerance can also be set for the voltage ranges using drive parameter p5040.

**Note**

The voltage range ± voltage tolerance must not overlap.

Speed limits

For the clamped states with state values 3 - 10, speed limit values can be specified using drive parameter p5043[0...6]. In the other clamped states (state values 1, 2 and 11), a limit value of 0 [rpm] permanently applies.

In the various clamped states, the controller limits the spindle speed to the applicable limit.

**Note**

Changing the speed limits

A change of the speed limits in drive parameter p5043[0...6] is only effective in the controller (limitation of the spindle speed to the new speed limit) after:

- Power-on reset or when the controller is switched-off/switched-on
- Deselection of the "Parking" state for the spindle (see Section "Parking a machine axis (Page 113)"
### Context: State value, voltage range and speed limit

<table>
<thead>
<tr>
<th>State value ¹)</th>
<th>Clamped state</th>
<th>Voltage range ²)</th>
<th>Speed limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sensor S1 not available or state values inactive</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>State initialization running</td>
<td>-</td>
<td>- ³)</td>
</tr>
<tr>
<td>2</td>
<td>Released with signal (error state)</td>
<td>-</td>
<td>p5041[0] + p5040 ³)</td>
</tr>
<tr>
<td>3</td>
<td>Released</td>
<td>p5041[0]</td>
<td>p5041[1]</td>
</tr>
<tr>
<td>4</td>
<td>Clamping with tool</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Releasing with tool</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>Releasing without tool</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>Clamped with tool AND S4 == 1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>Clamping without tool</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

¹) The state value can be read into the controller using the following system data:
- System variable: $VA_MOT_CLAMPING_STATE[<axis>]$
- NC/PLC interface: DB31, ... DBW134
- OPI variables: vaMotClampingState
- Drive parameters: r5001

²) p5041[0...5]: Voltage threshold values, p5040: Voltage threshold values tolerance

³) Speed limit permanently set: 0 [rpm]

### 16.9.4 Additional drive parameters

**P5042: Transition time**

The following times can be set in drive parameter p5042 for the clamped state identification:

- p5042[0]: Stabilization time for "clamped with tool"
  - The clamped state "clamped with tool" must be present in the spindle motor for at least the set stabilization time before the state is signaled to the controller.

- p5042[1]: Maximum time for clamping
  - The transition from the "released" state to the "clamped with tool" or "clamped without tool" state may take – as a maximum – the set time.

**r5044: Speed limitation from the clamping cycle**

The speed limit from p5043[6] which is active in the clamped state "clamped without tool" is displayed in drive parameter r5044.

A value of 65535 means that the speed limit is not active.
16.10 Supplementary conditions

16.10.1 Changing control parameters

For spindles that are not in position-controlled mode, machine data changes also take effect when the spindle is not stationary with the NEWCONF command.

In the case of changes to control parameters, speed setpoint jumps may occur when the new values take effect. Control parameters are, for example:

- MD32200 $MA_POSCTRL_GAIN (servo gain factor)
- MD32210 $MA_POSCTRL_INTEGR_TIME (position controller integral time)
- MD32410 $MA_AX_JERK_TIME (time constant for the axial jerk filter)

16.11 Examples

16.11.1 Automatic gear step selection (M40)

Example

To illustrate the contents of the new block search variables:

Assumptions about automatic gear step selection (M40):

S0...500 1. Gear step
S501..1000 2. Gear step
S1001..2000 3. Gear step

Content of system variables:

&P_SEARCH_S ; Collected S value
&P_SEARCH_DIR ; Collected direction of rotation
&P_SEARCH_GEAR ; Collected gear step

<table>
<thead>
<tr>
<th>Collected</th>
<th>S value:</th>
<th>Direction of rotation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>N05 G94 M40 M3</td>
<td>1000</td>
<td>3</td>
</tr>
<tr>
<td>S1000</td>
<td>3</td>
<td>40</td>
</tr>
<tr>
<td>N10 G96 S222</td>
<td>222</td>
<td>3</td>
</tr>
<tr>
<td>N20 G97</td>
<td>f (PlanAxPosPCS)* 3</td>
<td>40</td>
</tr>
<tr>
<td>N30 S1500</td>
<td>1500</td>
<td>3</td>
</tr>
<tr>
<td>N40 SPOS=0**</td>
<td>1500</td>
<td>-19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
</tr>
</tbody>
</table>
S1: Spindles

16.12 Data lists

16.12.1 Machine data

16.12.1.1 NC-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10192</td>
<td>GEAR_CHANGE_WAIT_TIME</td>
<td>Wait time for acknowledgment of a gear stage change during reorganization</td>
</tr>
<tr>
<td>10714</td>
<td>M_NO_FCT_EOP</td>
<td>M function for spindle active after <strong>RESET</strong></td>
</tr>
<tr>
<td>12060</td>
<td>OVR_SPIND_IS_GRAY_CODE</td>
<td>Spindle override with Gray coding</td>
</tr>
<tr>
<td>12070</td>
<td>OVR_FACTOR_SPIND_SPEED</td>
<td>Evaluation of spindle speed override switch</td>
</tr>
<tr>
<td>12080</td>
<td>OVR_REFERENCE_IS_PROG_FEED</td>
<td>Override reference velocity</td>
</tr>
<tr>
<td>12082</td>
<td>OVR_REFERENCE_IS_MIN_FEED</td>
<td>Defining the reference of the path override</td>
</tr>
<tr>
<td>12090</td>
<td>OVR_FUNCTION_MASK</td>
<td>Selection of override specifications</td>
</tr>
</tbody>
</table>

16.12.1.2 Channelspecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20090</td>
<td>SPIND_DEF_MASTER_SPIND</td>
<td>Initial setting for master spindle on channel</td>
</tr>
<tr>
<td>20092</td>
<td>SPIND_ASSIGN_TAB_ENABLE</td>
<td>Enabling/disabling of spindle converter</td>
</tr>
<tr>
<td>20850</td>
<td>SPOS_TO_VDI</td>
<td>Output of M19 to the PLC with SPOS/SPOSA</td>
</tr>
<tr>
<td>22400</td>
<td>S_VALUES_ACTIVE_AFTER_RESET</td>
<td>S function active after <strong>RESET</strong></td>
</tr>
</tbody>
</table>

* f (PlanAxPosPCS): The speed depends on the current position of the transverse axis in the workpiece coordinate system.

** ($P_SEARCH_SPOS and $P_SEARCH_SPOSMODE are programmed)
### 16.12.1.3 Axis/spindlespecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA_</th>
</tr>
</thead>
<tbody>
<tr>
<td>30300</td>
<td>IS_ROT_AX</td>
</tr>
<tr>
<td>30310</td>
<td>ROT_IS_MODULO</td>
</tr>
<tr>
<td>31044</td>
<td>ENC_IS_DIRECT2</td>
</tr>
<tr>
<td>31050</td>
<td>DRIVE_AX_RATIO_DENOM</td>
</tr>
<tr>
<td>31060</td>
<td>DRIVE_AX_RATIO_NUMERA</td>
</tr>
<tr>
<td>31064</td>
<td>DRIVE_AX_RATIO2_DENOM</td>
</tr>
<tr>
<td>31066</td>
<td>DRIVE_AX_RATIO2_NUMERA</td>
</tr>
<tr>
<td>31070</td>
<td>DRIVE_ENC_RATIO_DENOM</td>
</tr>
<tr>
<td>31080</td>
<td>DRIVE_ENC_RATIO_NUMERA</td>
</tr>
<tr>
<td>31122</td>
<td>BER0_DELAY_TIME_PLUS</td>
</tr>
<tr>
<td>31123</td>
<td>BER0_DELAY_TIME_MINUS</td>
</tr>
<tr>
<td>32200</td>
<td>POSCTRL_GAIN</td>
</tr>
<tr>
<td>32800</td>
<td>EQUIV_CURRCTRL_TIME</td>
</tr>
<tr>
<td>32810</td>
<td>EQUIV_SPEEDCTRL_TIME</td>
</tr>
<tr>
<td>32910</td>
<td>DYN_MATCH_TIME</td>
</tr>
<tr>
<td>34040</td>
<td>REFP_VELO_SEARCH_MARKER</td>
</tr>
<tr>
<td>34060</td>
<td>REFP_MAX_MARKER_DIST</td>
</tr>
<tr>
<td>34080</td>
<td>REFP_MOVE_DIST</td>
</tr>
<tr>
<td>34090</td>
<td>REFP_MOVE_DIST_CORR</td>
</tr>
<tr>
<td>34100</td>
<td>REFP_SET_POS</td>
</tr>
<tr>
<td>34200</td>
<td>ENC_REFP_MODE</td>
</tr>
<tr>
<td>35000</td>
<td>SPIND_ASSIGN_TO_MACHAX</td>
</tr>
<tr>
<td>35010</td>
<td>GEAR_STEP_CHANGE_ENABLE</td>
</tr>
<tr>
<td>35012</td>
<td>GEAR_STEP_CHANGE_POSITION</td>
</tr>
<tr>
<td>35014</td>
<td>GEAR_STEP_USED_IN_AXISMODE</td>
</tr>
<tr>
<td>35020</td>
<td>SPIND_DEFAULT_MODE</td>
</tr>
<tr>
<td>35030</td>
<td>SPIND_DEFAULT_ACT_MASK</td>
</tr>
<tr>
<td>35035</td>
<td>SPIND_FUNCTION_MASK</td>
</tr>
<tr>
<td>35040</td>
<td>SPIND_ACTIVE_AFTER_RESET</td>
</tr>
<tr>
<td>35090</td>
<td>NUM_GEAR_STEPS</td>
</tr>
<tr>
<td>35092</td>
<td>NUM_GEAR_STEPS2</td>
</tr>
<tr>
<td>35100</td>
<td>SPIND_VELO_LIMIT</td>
</tr>
<tr>
<td>35110</td>
<td>GEAR_STEP_MAX_VELO[n]</td>
</tr>
<tr>
<td>35112</td>
<td>GEAR_STEP_MAX_VELO2[n]</td>
</tr>
<tr>
<td>35120</td>
<td>GEAR_STEP_MIN_VELO[n]</td>
</tr>
</tbody>
</table>

- **Number**: The unique identifier for the machine axis or component.
- **Identifier: $MA_**: The specific identifier assigned to the axis or component.
- **Description**: A concise description of the function or parameter.
<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>35122</td>
<td>GEAR_STEP_MIN_VELO2[n]</td>
<td>2nd gear step data set: Minimum speed for automatic gear stage change</td>
</tr>
<tr>
<td>35130</td>
<td>GEAR_STEP_MAX_VELO_LIMIT[n]</td>
<td>Maximum speed of gear step</td>
</tr>
<tr>
<td>35135</td>
<td>GEAR_STEP_PC_MAX_VELO_LIMIT[n]</td>
<td>Maximum speed of gear step in position control</td>
</tr>
<tr>
<td>35140</td>
<td>GEAR_STEP_MIN_VELO_LIMIT[n]</td>
<td>Minimum speed of gear step</td>
</tr>
<tr>
<td>35150</td>
<td>SPIND_DES_VELO_TOL</td>
<td>Spindle speed tolerance</td>
</tr>
<tr>
<td>35160</td>
<td>SPIND_EXTERN_VELO_LIMIT</td>
<td>Spindle speed limitation via PLC</td>
</tr>
<tr>
<td>35200</td>
<td>GEAR_STEP_SPEEDCTRL_ACCEL[n]</td>
<td>Acceleration in speed control mode</td>
</tr>
<tr>
<td>35210</td>
<td>GEAR_STEP_POSCTRL_ACCEL[n]</td>
<td>Acceleration in position control mode</td>
</tr>
<tr>
<td>35212</td>
<td>GEAR_STEP_POSCTRL_ACCEL2[n]</td>
<td>2nd gear step data set: Acceleration in position control mode</td>
</tr>
<tr>
<td>35220</td>
<td>ACCEL_REDUCTION_SPEED_POINT</td>
<td>Speed limit for reduced acceleration</td>
</tr>
<tr>
<td>35230</td>
<td>ACCEL_REDUCTION_FACTOR</td>
<td>Reduced acceleration</td>
</tr>
<tr>
<td>35300</td>
<td>SPIND_POSCTRL_VELO</td>
<td>position control activation speed</td>
</tr>
<tr>
<td>35350</td>
<td>SPIND_POSITIONING_DIR</td>
<td>Positioning direction of rotation for a nonsynchronized spindle</td>
</tr>
<tr>
<td>35400</td>
<td>SPIND_OSCILL_DES_VELO</td>
<td>Oscillation speed</td>
</tr>
<tr>
<td>35410</td>
<td>SPIND_OSCILL_ACCEL</td>
<td>Oscillation acceleration</td>
</tr>
<tr>
<td>35430</td>
<td>SPIND_OSCILL_START_DIR</td>
<td>Oscillation start direction</td>
</tr>
<tr>
<td>35440</td>
<td>SPIND_OSCILL_TIME_CW</td>
<td>Oscillation time for M3 direction</td>
</tr>
<tr>
<td>35450</td>
<td>SPIND_OSCILL_TIME_CCW</td>
<td>Oscillation time for M4 direction</td>
</tr>
<tr>
<td>35500</td>
<td>SPIND_ON_SPEED_AT_IPO_START</td>
<td>Feed enable with spindle in setpoint range</td>
</tr>
<tr>
<td>35510</td>
<td>SPIND_STOPPED_AT_IPO_START</td>
<td>Feed enable with stationary spindle</td>
</tr>
<tr>
<td>35550</td>
<td>DRILL_VELO_LIMIT[n]</td>
<td>Maximum speeds for tapping</td>
</tr>
<tr>
<td>35590</td>
<td>PARAMSET_CHANGE_ENABLE</td>
<td>Parameter set definition possible from PLC</td>
</tr>
<tr>
<td>36060</td>
<td>STANDESTALL_VELO_TOL</td>
<td>Threshold velocity &quot;Axis/spindle stationary&quot;</td>
</tr>
<tr>
<td>36200</td>
<td>AX_VELO_LIMIT</td>
<td>Threshold value for velocity monitoring.</td>
</tr>
</tbody>
</table>

### 16.12.2 Setting data

### 16.12.2.1 Channelspecific setting data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $SC_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>42600</td>
<td>JOG_FEED_PER_REF_SOURCE</td>
<td>Revolutionary feedrate control in JOG mode</td>
</tr>
<tr>
<td>42800</td>
<td>SPIND_ASSIGN_TAB</td>
<td>Spindle number converter</td>
</tr>
<tr>
<td>42900</td>
<td>MIRROR_TOOL_LENGTH</td>
<td>Mirror tool length offset</td>
</tr>
<tr>
<td>42910</td>
<td>MIRROR_TOOL_WEAR</td>
<td>Mirror wear values of tool length compensation</td>
</tr>
<tr>
<td>42920</td>
<td>WEAR_SIGN_CUTPOS</td>
<td>Mirror wear values of machining plane</td>
</tr>
</tbody>
</table>
16.12.2  Axis/spindle-specific setting data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $SA_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>43200</td>
<td>SPIND_S</td>
<td>Speed for spindle start via PLC interface</td>
</tr>
<tr>
<td>43202</td>
<td>SPIND_CONSTCUT_S</td>
<td>Cutting rate for spindle start via PLC interface</td>
</tr>
<tr>
<td>43206</td>
<td>SPIND_SPEED_TYPE</td>
<td>For spindle start via PLC interface</td>
</tr>
<tr>
<td>43210</td>
<td>SPIND_MIN_VELO_G25</td>
<td>Progr. Spindle speed limiting G25</td>
</tr>
<tr>
<td>43220</td>
<td>SPIND_MAX_VELO_G26</td>
<td>Progr. Spindle speed limiting G26</td>
</tr>
<tr>
<td>43230</td>
<td>SPIND_MAX_VELO_LIMS</td>
<td>Progr. spindle speed limiting G96/G961</td>
</tr>
<tr>
<td>43235</td>
<td>SPIND_USER_VELO_LIMIT</td>
<td>Maximum spindle speed</td>
</tr>
<tr>
<td>43240</td>
<td>M19_SPOS</td>
<td>Spindle position for spindle positioning with M19</td>
</tr>
<tr>
<td>43250</td>
<td>M19_SPOSMODE</td>
<td>Spindle positioning approach mode for spindle positioning with M19</td>
</tr>
<tr>
<td>43300</td>
<td>ASSIGN_FEED_PER_REF_SOURCE</td>
<td>Rotational feedrate for positioning axes/spindles</td>
</tr>
</tbody>
</table>

16.12.3  signals

16.12.3.1  Signals to axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feedrate override A to H</td>
<td>DB31, ..., DBX0.0-7</td>
<td>DB380x.DBX0.0-7</td>
</tr>
<tr>
<td>Axis/spindle disable</td>
<td>DB31, ..., DBX1.3</td>
<td>DB380x.DBX1.3</td>
</tr>
<tr>
<td>Follow-up mode</td>
<td>DB31, ..., DBX1.4</td>
<td>DB380x.DBX1.4</td>
</tr>
<tr>
<td>Position measuring system 1</td>
<td>DB31, ..., DBX1.5</td>
<td>DB380x.DBX1.5</td>
</tr>
<tr>
<td>Position measuring system 2</td>
<td>DB31, ..., DBX1.6</td>
<td>DB380x.DBX1.6</td>
</tr>
<tr>
<td>Override active</td>
<td>DB31, ..., DBX1.7</td>
<td>DB380x.DBX1.7</td>
</tr>
<tr>
<td>Controller enable</td>
<td>DB31, ..., DBX2.1</td>
<td>DB380x.DBX2.1</td>
</tr>
<tr>
<td>Spindle reset/delete distancetogo</td>
<td>DB31, ..., DBX2.2</td>
<td>DB380x.DBX2.2</td>
</tr>
<tr>
<td>Velocity/spindle speed limitation</td>
<td>DB31, ..., DBX3.6</td>
<td>DB380x.DBX3.6</td>
</tr>
<tr>
<td>Program test Axis/Spindle Enable</td>
<td>DB31, ..., DBX3.7</td>
<td>DB380x.DBX3.7</td>
</tr>
<tr>
<td>Actual gear stage A to C</td>
<td>DB31, ..., DBX16.0-2</td>
<td>DB380x.DBX2000.0-2</td>
</tr>
<tr>
<td>Gear is changed over</td>
<td>DB31, ..., DBX16.3</td>
<td>DB380x.DBX2000.3</td>
</tr>
<tr>
<td>Resynchronize spindle 1</td>
<td>DB31, ..., DBX16.4</td>
<td>DB380x.DBX2000.4</td>
</tr>
<tr>
<td>Resynchronize spindle 2</td>
<td>DB31, ..., DBX16.5</td>
<td>DB380x.DBX2000.5</td>
</tr>
<tr>
<td>no n-monitoring with gear change</td>
<td>DB31, ..., DBX16.6</td>
<td>DB380x.DBX2000.6</td>
</tr>
</tbody>
</table>
### 16.12.3.2 Signals from axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle/No Axis</td>
<td>DB31, ... .DBX60.0</td>
<td>DB390x.DBX0.0</td>
</tr>
<tr>
<td>Encoder limit frequency exceeded 1</td>
<td>DB31, ... .DBX60.2</td>
<td>DB390x.DBX0.2</td>
</tr>
<tr>
<td>Encoder limit frequency exceeded 2</td>
<td>DB31, ... .DBX60.3</td>
<td>-</td>
</tr>
<tr>
<td>Homed/synchronized 1</td>
<td>DB31, ... .DBX60.4</td>
<td>DB390x.DBX0.4</td>
</tr>
<tr>
<td>Homed/synchronized 2</td>
<td>DB31, ... .DBX60.5</td>
<td>DB390x.DBX0.5</td>
</tr>
<tr>
<td>Position reached with exact stop coarse</td>
<td>DB31, ... .DBX60.6</td>
<td>DB390x.DBX0.6</td>
</tr>
<tr>
<td>Position reached with exact stop fine</td>
<td>DB31, ... .DBX60.7</td>
<td>DB390x.DBX0.7</td>
</tr>
<tr>
<td>Axis/spindle stationary (n &lt; n&lt;sub&gt;min&lt;/sub&gt;)</td>
<td>DB31, ... .DBX61.4</td>
<td>DB390x.DBX1.4</td>
</tr>
<tr>
<td>Position controller active</td>
<td>DB31, ... .DBX61.5</td>
<td>DB390x.DBX1.5</td>
</tr>
<tr>
<td>Speed controller active</td>
<td>DB31, ... .DBX61.6</td>
<td>DB390x.DBX1.6</td>
</tr>
<tr>
<td>Current controller active</td>
<td>DB31, ... .DBX61.7</td>
<td>DB390x.DBX1.7</td>
</tr>
<tr>
<td>Restored 1</td>
<td>DB31, ... .DBX71.4</td>
<td>DB390x.DBX11.4</td>
</tr>
<tr>
<td>Restored 2</td>
<td>DB31, ... .DBX71.5</td>
<td>DB390x.DBX11.5</td>
</tr>
<tr>
<td>Setpoint gear stage A to C</td>
<td>DB31, ... .DBX82.0-2</td>
<td>DB390x.DBX2000.0-2</td>
</tr>
<tr>
<td>Change gear</td>
<td>DB31, ... .DBX82.3</td>
<td>DB390x.DBX2000.3</td>
</tr>
<tr>
<td>Speed limit exceeded</td>
<td>DB31, ... .DBX83.0</td>
<td>DB390x.DBX2001.0</td>
</tr>
<tr>
<td>Setpoint speed limited</td>
<td>DB31, ... .DBX83.1</td>
<td>DB390x.DBX2001.1</td>
</tr>
<tr>
<td>Setpoint speed increased</td>
<td>DB31, ... .DBX83.2</td>
<td>DB390x.DBX2001.2</td>
</tr>
<tr>
<td>Spindle in setpoint range</td>
<td>DB31, ... .DBX83.5</td>
<td>DB390x.DBX2001.5</td>
</tr>
<tr>
<td>Actual direction of rotation clockwise</td>
<td>DB31, ... .DBX83.7</td>
<td>DB390x.DBX2001.7</td>
</tr>
<tr>
<td>Rigid tapping active</td>
<td>DB31, ... .DBX84.3</td>
<td>DB390x.DBX2002.3</td>
</tr>
<tr>
<td>active spindle mode synchronous mode</td>
<td>DB31, ... .DBX84.4</td>
<td>DB390x.DBX2002.4</td>
</tr>
<tr>
<td>Active spindle positioning mode</td>
<td>DB31, ... .DBX84.5</td>
<td>DB390x.DBX2002.5</td>
</tr>
<tr>
<td>Active spindle mode oscillation mode</td>
<td>DB31, ... .DBX84.6</td>
<td>DB390x.DBX2002.6</td>
</tr>
<tr>
<td>Active spindle control mode</td>
<td>DB31, ... .DBX84.7</td>
<td>DB390x.DBX2002.7</td>
</tr>
</tbody>
</table>
## S1: Spindles

**16.12 Data lists**

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spindle actually reached in position</td>
<td>DB31, ... .DBX85.5</td>
<td>DB390x.DBX2003.5</td>
</tr>
<tr>
<td>M function for spindle</td>
<td>DB31, ... .DBB86-87</td>
<td>DB370x.DBDO000</td>
</tr>
<tr>
<td>S function for spindle</td>
<td>DB31, ... .DBB88-91</td>
<td>DB370x.DBD0004</td>
</tr>
<tr>
<td>Sensors available</td>
<td>DB31, ... .DBX132.0</td>
<td>DB390x.DBX7000.0</td>
</tr>
<tr>
<td>Sensor S1 available (clamped state)</td>
<td>DB31, ... .DBX132.1</td>
<td>DB390x.DBX7000.1</td>
</tr>
<tr>
<td>Sensor S4 available (piston end position)</td>
<td>DB31, ... .DBX132.4</td>
<td>DB390x.DBX7000.4</td>
</tr>
<tr>
<td>Sensor S5 available (angular position of the motor shaft)</td>
<td>DB31, ... .DBX132.5</td>
<td>DB390x.DBX7000.5</td>
</tr>
<tr>
<td>State value is generated, speed limitation p5043 is active</td>
<td>DB31, ... .DBX133.2</td>
<td>DB390x.DBX7001.2</td>
</tr>
<tr>
<td>Clamped state</td>
<td>DB31, ... .DBW134</td>
<td>DB390x.DBW7002</td>
</tr>
<tr>
<td>Analog value: Clamped state</td>
<td>DB31, ... .DBW136</td>
<td>DB390x.DBW7004</td>
</tr>
<tr>
<td>Sensor S4 (piston end position)</td>
<td>DB31, ... .DBX138.4</td>
<td>DB390x.DBX7006.4</td>
</tr>
<tr>
<td>Sensor S5 (angular position of the motor shaft)</td>
<td>DB31, ... .DBX138.5</td>
<td>DB390x.DBX7006.5</td>
</tr>
</tbody>
</table>
17.1 Brief description

Types of feedrate

The feedrate determines the machining speed (axis or path velocity) and is observed in every type of interpolation, even where allowance is made for tool offsets on the contour or on the tool center point path (depending on G functions).

The following types of feedrate allow optimum adaptation to the various technological applications (turning, milling, drilling, etc.):

- Rapid traverse feedrate (G0)
- Inverse-time feedrate (G93)
- Linear feedrate (G94)
- Revolutionary feedrate (G95)
- Constant cutting rate (G96, G961)
- Constant speed (G97, G971)
- Feedrate for thread cutting (G33, G34, G35)
- Feedrate for tapping without compensating chuck (G331, G332)
- Feedrate for tapping with compensating chuck (G63)
- Feedrate for chamfer/rounding FRC, FRCM
- Non-modal feedrate FB

Axis assignment of the feedrates

Feedrates can be assigned to the axes variably to adjust to the different technological requirements.

The following versions are possible:

- Separate feedrates for the working plane and the infeed axis
- Variable axis assignment for path feedrate
- Feedrate for positioning axes

Feedrate control

The programmed feedrate can be changed during the machining or for test purposes to enable adjustment to the changed technological conditions.

- Via the machine control panel
- Via the operator panel front
17.2 Path feedrate $F$

Path feedrate $F$

The path feedrate represents the geometrical total of the speed components in the participating axes. It is therefore generated from the individual motions of the interpolating axes.

The default uses the axial speeds of the geometry axes which have been programmed. The `FGROUP` command can be used to include other geometry and/or synchronized axes in the calculation of the path feedrate.

The path feedrate $F$ determines the machining speed and is observed in every type of interpolation even where allowance is made for tool offsets. The value programmed under the address $F$ remains in a program until a new $F$ value or a new type of feedrate is programmed.

Range of values for path feedrate $F$

See Description of Functions G2: "Speeds, Setpoint / Actual Value Systems, Closed-Loop Control", Section: "Velocities (Page 317)".

F value at PLC interface

The $F$ value of the current path feedrate is always entered in the channel-specific PLC interface for auxiliary functions (DB21, ... DBB158 to 193).

For explanations about the associated interface signals (change signal, $F$ value), see Description of Functions "H2: Auxiliary function outputs to PLC (Page 377)".

Feedrate with transition circle

References:
Programming Manual, Fundamentals

Feedrate for internal radius and external radius path sections

For circular blocks or spline blocks with curvature in the same direction and tool radius offset activated (G41/G42), the programmed feedrate can act on the center point path or on the contour (depending on the internal radius or external radius path sections).
A group of G functions is provided for this purpose:

- **CFTCP**
  Programmed feedrate acting on the center point path.

- **CFC**
  Programmed feedrate acting on the contour.

- **CFCIN**
  Programmed feedrate acting only on the contour with a concave spline.

**References:**
Programming Manual, Fundamentals

**Maximum tool path velocity**

The maximum path velocity results from the maximum velocities of the linear or rotary axes involved (MD32000 $MA_MAX_AX_VEL0), i.e. the axis with the lowest maximum velocity determines the maximum path velocity. This cannot be exceeded.

If G0 is programmed, traversing is at the path velocity resulting from the MD32000 $MA_MAX_AX_VEL0 limitation.

**Limit velocity for path axes**

In addition, the FL[<axis>] statement can be used to program a limit velocity for path axes (geometry and synchronized axes).

This enables separate feedrates to be programmed for the working plane and infeed axis. This means that a feedrate is specified for the path-related interpolation and for the infeed axis. The axis perpendicular to the selected machining plane is designated as the infeed axis. The infeed axis-specific feedrate can be programmed to limit the axis velocity and therefore the path velocity. No coordinate rotations through frames should be included, i.e. the infeed axis must be an axis of the standard coordinate system. This function can be used to compensate for the fact that a cutter has a lower cutting performance on the face side than across the cutter circumference.

Programming example:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>... G94 ...</td>
<td>; Selection of feedrate type (mm/min)</td>
</tr>
<tr>
<td>X30 Y20 F200</td>
<td>; Path feedrate = 200 mm/s</td>
</tr>
<tr>
<td>FL[Z]=50 Z=30</td>
<td>; Max. feedrate for Z axis: 50 mm/s</td>
</tr>
</tbody>
</table>

**Low-resolution encoders**

When using low-resolution encoders, more continuous path or axis motions can be achieved with smoothed actual values. The larger the time constant, the better the smoothing of the actual values, and the longer the overtravel.

MD34990 $MA_ENC_ACTVAL_SMOOTH_TIME[<axis>] (smoothing time constant for actual values)
17.2.1  Feedrate type G93, G94, G95

Effectiveness

The feedrate types G93, G94, G95 are active for the G functions of group 1 (except G0) in the automatic modes.

G94 or G95 can be used for traversing in JOG mode.

References:
Function Manual, Extended Functions; Manual and Handwheel Travel (H1)

Inverse-time feedrate (G93)

The inverse-time feedrate is used when it is easier to program the duration, rather than the feedrate, for retraction of a block.

The inverse-time feedrate is calculated from the following formula:

\[
F = \frac{v}{s}
\]

with
- \( F \): Inverse-time feedrate in rpm
- \( v \): Required path velocity in mm/min or inch/min
- \( s \): Path length in mm/inch

Programming example:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G1 G93 X100 Y200 F2</td>
<td>; The programmed path is traversed in 0.5 min.</td>
</tr>
<tr>
<td>…</td>
<td></td>
</tr>
</tbody>
</table>

Note

G93 may not be used when G41/G42 is active. If the block length varies greatly from block to block, a new F value should be programmed in each block for G93.
Linear feedrate (G94)

The linear feedrate is programmed in the following units relative to a linear or rotary axis:

- [mm/min, degrees/min] on standard metric systems
- [inch/min, degrees/min] on standard imperial systems

Revolutional feedrate (G95)

The revolutional feedrate is programmed in the following units relative to a master spindle:

- [mm/rev] on standard metric systems
- [inch/rev] on standard imperial systems
- [degrees/rev] on a rotary axis

The path velocity is calculated from the actual speed of the spindle according to the following formula:

\[ V = n \times F \]

with
- \( V \): Path velocity in mm/min or inch/min
- \( n \): Speed of the master spindle in rpm
- \( F \): Programmed revolutional feedrate in mm/rev or inch/rev

Note

The programmed \( F \) value is deleted when the system switches between the feedrate types G93, G94 and G95.

Tooth feedrate

Primarily for milling operations, the tooth feedrate \( FZ \ldots \) (feed distance per tooth), which is more commonly used in practice, can be programmed instead of the revolutional feedrate \( F \ldots \):

The controller uses the \( $TC\_DPNT \) (number of teeth per revolution) tool parameter associated with the active tool offset data record to calculate the effective revolutional feedrate for each traversing block from the programmed tooth feedrate.

\[ F = FZ \times $TC\_DPNT \]

with
- \( F \): Revolutional feedrate in mm/rev or inch/rev
- \( FZ \): Tooth feedrate in mm/tooth or inch/tooth
- \( $TC\_DPNT \): Tool parameter: Number of teeth/rev
Example: Milling cutter with 5 teeth ($TC_{DPNE} = 5$)

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G0 X100 Y50</td>
<td>; Tool feedrate 0.02 mm/tooth</td>
</tr>
<tr>
<td>N20 G1 G95 FZ=0.02</td>
<td>; Load tool and activate tool offset data record.</td>
</tr>
<tr>
<td>N30 T3 D1</td>
<td>; Spindle speed 200 rpm</td>
</tr>
<tr>
<td>N40 M3 S200</td>
<td>; Milling with:</td>
</tr>
<tr>
<td>N50 X20</td>
<td>→ effective revolutional feedrate:</td>
</tr>
<tr>
<td></td>
<td>F = 0.02 mm/tooth * 5 teeth/rev = 0.1 mm/rev</td>
</tr>
<tr>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td>F = 0.1 mm/rev * 200 rpm = 20 mm/min</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Revolutional feedrate in JOG mode**

In JOG mode, the response of the axis/spindle also depends on the following setting data:

SD41100 $SN_{_JOG\_REV\_IS\_ACTIVE}$ (revolutional feedrate for JOG active)

If this setting data is active, an axis/spindle is always moved with revolutional feedrate:

MD32050 $MA_{_JOG\_REV\_VELO}$ (revolutional feedrate with JOG)

or

MD32040 $MA_{_JOG\_REV\_VELO\_RAPID}$ (revolutional feedrate with JOG with rapid traverse overlay)

depending on the master spindle.

If the setting data is inactive:

- The response of the axis/spindle depends on the setting data:
  SD43300 $SA_{_ASSIGN\_FEED\_PER\_REV\_SOURCE}$ (revolutional feedrate for position axes/spindles)

- The response of a geometry axis on which a frame acts is to rotate, depending on the channel-specific setting data:
  SD42600 $SC_{_JOG\_FEED\_PER\_REV\_SOURCE}$

**DB31, ... DBX62.2 (revolutional feedrate active)**

A programmed, active revolutional feedrate (G95) is displayed using the following NC/PLC interface signal.

DB31, ... DBX62.2 (revolutional feedrate active)

### 17.2.2 Type of feedrate G96, G961, G962, G97, G971

**Constant cutting rate (G96, G961)**

The constant cutting rate is used on turning machines to keep the cutting conditions constant, independently of the work diameter of the workpiece. This allows the tool to be operated in the optimum cutting performance range and therefore increases its service life.
Selection of G96, G961:
When programming G96, G961, the corresponding S value is interpreted as the cutting rate in m/min or ft/min along the transverse axis. If the workpiece diameter decreases during machining, the speed is increased until the constant cutting speed is reached.

When G96, G961 is first selected in the part program, a constant cutting rate must be entered in mm/min or ft/min.

With G96, the control system will automatically switch to revolutional feedrate (as with G95), i.e. the programmed feedrate F is interpreted in mm/rev or inch/rev.

When programming G961, linear feedrate is selected automatically (as with G94). A programmed feedrate F is interpreted in mm/min or inch/min.

Determining the spindle speed
Based on the programmed cutting rate (either $S_{G96}$ or $S_{G961}$) and the actual cartesian position of the transverse axis (radius), the control system calculates the spindle speed at the TCP using the following formula:

$$ n = \frac{S_{\text{Speed}}}{2 \cdot \pi \cdot r} $$

- $n$: Spindle speed
- $S_{\text{Speed}}$: Programmed cutting rate
- $\pi$: Circle constant
- $r$: Radius (distance, center of rotation to TCP)

The following is assumed when determining the radius:

- The transverse axis position 0 in the WCS represents the center of rotation.
- Position offsets (such as online tool offset, external zero offset, $AA_{OFF}$, DRF offset and compile cycles) and position components through couplings (e.g. following axis for TRAIL) are not taken into account when determining the radius.
- Frames (e.g. programmable frames such as SCALE, TRANS or ROT) are taken into account in the calculation of the spindle speed and can bring about a change in speed, if the effective diameter at the TCP changes.

Diameter programming and reference axis for several transverse axes in one channel:
One or more transverse axes are permitted and can be activated simultaneously or separately:

- Programming and displaying in the user interface in the diameter
- Assignment of the specified reference axis with $SCC[<axis>]$ for a constant cutting rate G96, G961, G962

For more information, see Description of Functions "P1: Transverse axes (Page 811)".
Example

\[ S_{G96} = 230 \, \text{m/min} \]

- Where \( r = 0.2 \, \text{m} \rightarrow n = 183.12 \, \text{rpm} \)
- Where \( r = 0.1 \, \text{m} \rightarrow n = 366.24 \, \text{rpm} \)

⇒ The smaller the workpiece diameter, the higher the speed.

For G96, G961 or G962, a geometry axis must be defined as the transverse axis.

The transverse axis, whose position affects the speed of the master spindle, is defined using channel-specific machine data:

MD20100 $MC_DIAMETER_AX_DEF (geometry axis with transverse axis function)

The function G96, G961 or G962 requires that the machine zero and the workpiece zero of the transverse axis are in the turning center of the spindle.

Constant speed (G97, G971)

G97, G971 deactivates the "Constant cutting rate function" (G96, G961) and saves the last calculated spindle speed. With G97, the feedrate is interpreted as a rotational feedrate (as with G95). When programming G971, linear feedrate is selected (as with G94). The feedrate \( F \) is interpreted in mm/min or inch/min.

When G97, G971 is active, an \( S \) value can be reprogrammed to define a new spindle speed. This will not modify the cutting rate programmed in G96, G961.

G97, G971 can be used to avoid speed variations in motions along the transverse axis without machining (e.g. cutting tool).

Note

G96, G961 is only active during workpiece machining (G1, G2, G3, spline interpolation, etc., where feedrate \( F \) is active).

The response of the spindle speed for active G96, G961 and G0 blocks can be defined in the channel-specific machine data:

MD20750 ALLOW_G0_IN_G96 (G0 logic for G96, G961)

When constant cutting rate G96, G961 is selected, no gear stage change can take place.

The spindle speed override switch acts on the spindle speed calculated.

A DRF offset in the transverse axis does not affect the spindle speed setpoint calculation.

At the start of machining (after G0) and after NC stop, G60, G09, ... the path start waits for "nAct = nSet".

The interface signals "nAct = nSet" and "Set speed limited" are not modified by internal speed settings.

When the speed falls below the minimum speed or if the signal "Axis/spindle stationary" is recognized, "nAct = nSet" is reset.

A path operation which has started (G64, rounding), is not interrupted.
Spindle speed limitation with G96, G961

A maximum spindle speed can be specified for the "Constant cutting rate" function:

- In the setting data:
  SD43230 $SA_SPIND_MAX_VELO_LIMS (spindle speed limitation for G96/G961)
- In the part program (for the master spindle) with the programming command LIMS

The most recently changed value (LIMS or SD) is active.

LIMS is effective with G96, G961, G97 and can be specified on up to four speed limitations in the part program in one block. Spindle number \(<Sn> = 1, 2, 3, \text{ or } 4\) of the master spindle that is possible in the particular instance can be programmed in part program command LM[<Sn>].

When the block is loaded in the main run, all programmed values are transferred to the setting data SD43230 $SA_SPIND_MAX_VELO_LIMS.

Depending on the machine data:
MD10710 PROG_SD_RESET_SAVE_TAB[n] (setting data to be updated),
the speed limit set with LIMS remains stored after the control is switched off.
When G96, G961, G97 are reactivated, this speed limitation is also activated.

The maximum permissible spindle speed defined via G26 or via the setting data:
SD43220 $SA_SPIND_MAX_VELO_G26 (maximum spindle speed)
cannot be exceeded.

In the event of incorrect programming that would cause one of the speed limits (G26 or SD43220 $SA_SPIND_MAX_VELO_G26) to be exceeded, the following interface signal is set.
DB31, ... DBX83.1 (programmed speed too high)

In order to ensure smooth rotation with large part diameters, the spindle speed is not permitted to fall below a minimum level.
This speed can be set via the setting data:
SD43210 $SA_SPIND_MIN_VELO_G25 (minimum spindle speed)
and, depending on the gear step, with the machine data:
MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT (minimum speed of the gear step).

The minimum spindle speed can be changed in the part program with G25. In the event of incorrect programming that would cause one of the speed limits (G25 or SD43210 $SA_SPIND_MIN_VELO_G25) to be undershot, the following interface signal is set.
DB31, ... DBX83.2 (speed setpoint too low)
For more information on the spindle-speed limitations, see function description S1: "Spindles", Section: "Spindle monitoring (Page 1283)".

**Note**

The speed limits changed with G25/G26/LIMS in the part program are taken into the setting data and therefore remain saved after the end of program.

However, if the speed limits changed with G25/G26/LIMS are no longer to apply after the end of program, the following definitions must be inserted in the GUD block of the machine manufacturer:

```plaintext
REDEF $SA_SPIND_MIN_VELO_G25 PRLOC
REDEF $SA_SPIND_MAX_VELO_G26 PRLOC
REDEF $SA_SPIND_MAX_VELO_LIMS PRLOC
```

**Master spindle changeover with G96, G961**

If the master spindle is switched over when G96, G961 are active, the speed of the former master spindle is retained. This corresponds to a transition from G96 to G97. The master spindle newly defined with SETMS executes the "Constant cutting rate" function generated in this way.

**Alarms**

**Constant cutting rate G96, G961, G962**

- If no F value is programmed, alarm 10860 "No feedrate programmed" is output. The alarm is not generated with G0 blocks.
- Alarm 14800 "Programmed path velocity smaller than or equal to zero" is output while programming a negative path velocity.
- If, with an active G96, G961 or G962, no transverse axis is defined in the machine data: MD20100 $MC_DIAMETER_AX_DEF (geometry axis with transverse axis function), alarm 10870 "No transverse axis defined" is output.
- If a negative maximum spindle speed is programmed with the LIMS program command when G96, G961 are active, alarm 14820 "Negative maximum spindle speed programmed for G96, G961" is output.
- If no constant cutting rate is programmed when G96, G961 is selected for the first time, alarm 10900 "No S value programmed for constant cutting rate" is output.
17.2.3 Feedrate for thread cutting (G33, G34, G35, G335, G336)

17.2.3.1 Feedrate with G33

G33

The function G33 can be used to machine threads with constant pitch of the following type:

**Speed S, feedrate F, thread pitch**

A revolutional feedrate [mm/revolution] is used for G33 threads. The revolutional feedrate is defined by programming the thread pitch [mm/revolution].

The speed of the axes for the thread length is calculated from the programmed spindle speed $S$ and the thread pitch.

**Feedrate $F$ [mm/min] = speed $S$ [rev/min] * pitch [mm/rev]**

At the end of the acceleration ramp, the position coupling between the spindle actual value (spindle setpoint with SPCON on master spindle) and the axis setpoint is established. At this moment, the position of the axis in relation to the zero mark of the spindle (including zero mark offsets) is as if the axis had accelerated abruptly at the start of the block when the thread start position (zero mark plus SF) was crossed. Compensation is made for the following error of the axis.

**Minimum spindle speed**

In order to ensure smooth rotation at low speeds, the spindle speed is not permitted to fall below a minimum level.

This speed can be set:

- With the setting data:
  
  SD43210 SSA_SPIND_MIN_VELO_G25 (minimum spindle speed)

- For each gear stage with the machine data:
  
  MD35140 $MA\_GEAR\_STEP\_MIN\_VELO\_LIMIT$ (minimum speed for gear stage change)

The minimum spindle speed can be changed in the part program with G25.

**NC stop, single block**

NC stop and single block (even at the block boundary) are only active after completion of thread chaining. All successive G33 blocks and the first following non-G33 block are traversed as a block.

**Premature abortion without destruction**

Thread cutting can be aborted without destruction before the end point is reached. This can be done by activating a retraction motion.
Thread cutting with ROT frame

With ROT frame and G33, G34, G35, alarm 10607 "Thread with frame not executable" is activated if the rotation causes a change in the thread length and thus the pitch. Rotation around the thread axis is permissible.

Alarm 10607 "Thread with frame not executable" can be suppressed by setting bit 12 in machine data MD11410 $MN_SUPPRESS_ALARM_MASK, if the ROT instruction is used intentionally in the application.

All other frames are accepted by the NC without alarm. Attention is drawn to the pitch-changing effect of SCALE.

17.2.3.2 Programmable run-in and run-out path for G33, G34 and G35

Function

The run-in and run-out path of the thread can be specified with the DITS and DITE statements. The thread axis is accelerated or braked within the specified path.

![Diagram showing run-in and run-out paths]

① Run-in/run-out path, depending on the machining direction

Short run-in path

Due to the collar on the thread run-in, little room is left for the tool start ramp. This must therefore be specified shorter via DITS.

Short run-out path

Due to the collar on the thread run-out, little room is left for the deceleration ramp, leading to the risk of collision between the workpiece and the tool edge. The deceleration ramp can be specified shorter using DITE. Due to the inertia of the mechanical system, however, a collision can still occur.

Remedy: Program a shorter thread, reduce the spindle speed.

Note

DITE acts at the end of the thread as a rounding clearance. This achieves a smooth change in the axis motion.
Effects
The programmed run-in and run-out path only increases the rate of acceleration on the path. If one of the two paths is set larger than the thread axis needs with active acceleration, the thread axis is accelerated or decelerated with maximum acceleration.

Note
The axis can be overloaded if the specified path is too short.

Activation
The DITS and DITE statements are always active for thread cutting.

Example

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N...</td>
<td></td>
</tr>
<tr>
<td>N59 G90 G0 Z100 X10 SOFT M3 S500</td>
<td></td>
</tr>
<tr>
<td>N60 G33 Z50 K5 SF=180 DITS=1 DITE=3 ; Start of corner rounding with Z=53</td>
<td></td>
</tr>
<tr>
<td>N61 G0 X20</td>
<td></td>
</tr>
</tbody>
</table>

Note
Only paths, and not positions, are programmed with DITS and DITE.
The programmed run-in/run-out path is handled according to the current dimension setting (inches, metric).

Setting data SD42010 (acceleration behavior of axis when thread cutting)
When a block containing DITS and/or DITE is inserted in the main run, the programmed run-in/run-out path is taken into the setting data:

- SD42010 $SC_THREAD_RAMP_DISP[ 0 ] = programmed value of DITS
- SD42010 $SC_THREAD_RAMP_DISP[ 1 ] = programmed value of DITE

If no run-in/run-out path is programmed before or in the first thread block, the current value of the setting data is used.
SD42010 $SC_THREAD_RAMP_DISP[ 0 ] = <value>
17.2 Path feedrate F

17.2.3.3 Linear increasing/decreasing thread pitch change with G34 and G35

Function

The thread pitch increase (G34) defines the numerical increase in the pitch value. A larger pitch results in a larger distance between the threads on the workpieces. The velocity of the thread axis therefore increases with assumed constant spindle speed.

The opposite therefore applies for the decrease in thread pitch (G35).

The following definitions are made for the thread pitch change:

- G34: Increase in thread pitch corresponds to progressive change
- G35: Decrease in thread pitch corresponds to degressive change

Both G34 and G35 functions imply the functionality of G33 and also provide the option of programming an absolute pitch change value for the thread under F. If the start and end pitch of a thread is known, the thread pitch change can be determined using the following equation:

\[ F = \frac{|k_a^2 - k_e^2|}{2 \times l_G} \]

The meaning is as follows:

- \( F \): The thread pitch change to be programmed [mm/rev²]
- \( k_e \): Thread pitch of axis target point coordinate, thread axis [mm/rev]
- \( k_a \): Initial thread pitch (programmed under I, J or K) [mm/rev]
- \( l_G \): Thread length [mm]

The absolute value of \( F \) must be applied to G34 or G35 depending on the required pitch increase or decrease.
When the thread length $l$, pitch change $F$ and initial pitch $k_a$ are known, the pitch increase at the end of block $k_e$ can be determined as follows by modifying the formula:

- For G34 (increasing pitch):
  \[ k_e = \sqrt{k_a^2 + F \times 2 \times l} \]

- For G35 (decreasing pitch):
  \[ k_e = \sqrt{k_a^2 - F \times 2 \times l} \]

**Note**

If the formula results in a negative root expression, the thread cannot be machined!

In this case, the NC signals alarm 10605 or alarm 22275.

**Application**

The G34 and G35 functions can be used to produce self-shearing threads.

**Example**

Thread cutting G33 with decreasing thread pitch G35

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1608 M3 S10</td>
<td>; Spindle speed</td>
</tr>
<tr>
<td>N1609 G0 G64 Z40 X216</td>
<td>; Approach starting point</td>
</tr>
<tr>
<td>N1610 G33 Z0 K100 SF=R14</td>
<td>; Thread with constant pitch 100 mm/rev</td>
</tr>
<tr>
<td>N1611 G35 Z-220 K100 F17.045455</td>
<td>; Thread pitch decrease 17.045455 mm/rev2</td>
</tr>
<tr>
<td>N1612 G33 Z-240 K50</td>
<td>; Thread pitch at end of block 50 mm/rev</td>
</tr>
<tr>
<td>N1613 G0 X218</td>
<td>; Traverse thread block without jerk</td>
</tr>
<tr>
<td>N1614 G0 Z40</td>
<td></td>
</tr>
<tr>
<td>N1616 M17</td>
<td></td>
</tr>
</tbody>
</table>

**Monitoring during the block preparation**

Any pitch changes that would overload the thread axis when G34 is active or would result in an axis standstill when G35 is active, are detected in advance during block preparation. Alarm 10604 “Thread pitch increase too high” or 10605 “Thread pitch decrease too high” is signaled.

During thread cutting, certain practical applications require a correction of the spindle speed. In this case, the operator will base his correction on the permissible velocity of the thread axis.

To do this, it is possible to suppress the output of alarms 10604 and 10605 as follows:
MD11410 $MN_SUPPRESS_ALARM_MASK bit 10 = 1
Block preparation is then continued normally.

**Monitoring during the execution**

The following situations are monitored cyclically when the thread is machined (interpolation):

- Exceeding of maximum velocity of thread axis
- Reaching of axis standstill with G35

The following alarm is signaled when the monitoring function responds:

- Alarm 22269 "Maximum velocity of thread axis reached" or
- Alarm 22275 "Zero velocity of thread axis reached"

**17.2.3.4 Fast retraction during thread cutting**

**Function**

The "Fast retraction for thread cutting (G33)" function can be used to interrupt thread cutting without causing irreparable damage in the following circumstances:

- NC stop (NC/PLC interface signal)
- Alarms that implicitly trigger NC stop
- Switching of a rapid input

**References**

Programming Manual, Job Planning; Section "Fast retraction from the contour"

The retraction motion can be programmed via:

- Retraction path and retraction direction (relative)
- Retraction position (absolute)

**Note**

**Tapping**

The "Fast retraction" function cannot be used with tapping (G331/G332).

**Programming**

**Syntax**

Enable fast retraction, retraction motion via retraction path and retraction direction:

```plaintext
G33 ... LFON DILF=<value> LFTXT/LFWP ALF=<value>
```

Enable fast retraction, retraction motion via retraction position:

```plaintext
POLF[<axis name>]=<value> LFPOS
POLFMASK/POLFMLIN(<axis 1 name>,<axis 2 name>, etc.)
G33 ... LFON
```
Disable fast retraction for thread cutting:
LFOF

Meaning

LFON: Enable fast retraction for thread cutting (G33).
LFOF: Disable fast retraction for thread cutting (G33).
DILF: Define length of retraction path.
The value preset during MD configuration (MD21200 $MC_LIFTFAST_DIST) can be modified in the part program by programming DILF.

Note:
The configured MD value is always active following NC-RESET.

LFTXT
The retraction direction is controlled in conjunction with ALF with G functions LFTXT and LFWP.
LFTXT: The plane in which the retraction motion is executed is calculated from the path tangent and the tool direction (default setting).
LFWP: The plane in which the retraction motion is executed is the active working plane.

ALF: The direction is programmed in discrete degree increments with ALF in the plane of the retraction motion.
With LFTXT, retraction in the tool direction is defined for ALF=1.
With LFWP the direction in the working plane is derived from the following assignment:
• G17 (X/Y plane)
  ALF=1; Retraction in the X direction
  ALF=3; Retraction in the Y direction
• G18 (Z/X plane)
  ALF=1; Retraction in the Z direction
  ALF=3; Retraction in the X direction
• G19 (Y/Z plane)
  ALF=1; Retraction in the Y direction
  ALF=3; Retraction in the Z direction

References:
Programming options with ALF are also described in "Traverse direction for fast retraction from the contour" in the Programming Manual, Job Planning.

LFPOS: Retraction of the axis declared with POLFMASK or POLFMLIN to the absolute axis position programmed with POLF.
POLFMASK: Release of axes (<axis 1 name>,<axis 1 name>, etc.) for independent retraction to absolute position.
POLFMLIN: Release of axes for retraction to absolute position in linear relation

Note:
Depending on the dynamic response of all the axes involved, the linear relation cannot always be established before the lift position is reached.
POLF[]: Define absolute retraction position for the geometry axis or machine axis in the index
Effective: Modal
=<>value>: In the case of geometry axes, the assigned value is interpreted as a position in the workpiece coordinate system. In the case of machine axes, it is interpreted as a position in the machine coordinate system.
The values assigned can also be programmed as incremental dimensions:

=IC<>value>

<axis name>: Name of a geometry axis or machine axis

Note
LFON or LFOF can always be programmed, but the evaluation is performed exclusively during thread cutting (G33).

Note
POLF with POLFMASK/POLFMLIN are not restricted to thread cutting applications.

Figure 17-1 Interruption of G33 through retraction motion
Dynamic response of the retraction motion

The retraction motion is executed with maximum axis dynamic response:

- MD32000 $MA_MAX_AX_VELO[<axis>] (velocity)
- MD32300 $MA_MAX_AX_ACCEL[<axis>] (acceleration)
- MD32431 $MA_MAX_AX_JERK[<axis>] (jerk)

Example

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N55 M3 S500 G90 G18</td>
<td>; Set active machining plane.</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N65 MSG (&quot;thread cutting&quot;)</td>
<td></td>
</tr>
<tr>
<td>MM_THREAD:</td>
<td></td>
</tr>
<tr>
<td>N67 $AC_LIFTFAST=0</td>
<td>; Reset before starting the thread.</td>
</tr>
<tr>
<td>N68 G0 Z5</td>
<td></td>
</tr>
<tr>
<td>N69 X10</td>
<td></td>
</tr>
<tr>
<td>N70 G33 Z30 K5 LFON DILF=10 LFWP ALF=7</td>
<td>; Enable fast retraction for thread cutting.</td>
</tr>
<tr>
<td></td>
<td>; Retraction path - 10 mm</td>
</tr>
<tr>
<td></td>
<td>; Retraction plane Z/X (because of G18).</td>
</tr>
<tr>
<td></td>
<td>; Retraction direction -X (with ALF=3; retraction direction +X).</td>
</tr>
<tr>
<td>N71 G33 Z55 X15</td>
<td></td>
</tr>
<tr>
<td>N72 G1</td>
<td></td>
</tr>
<tr>
<td>N69 IF $AC_LIFTFAST GOTOB MM_THREAD</td>
<td>; If thread cutting has been interrupted.</td>
</tr>
<tr>
<td>N90 MSG (&quot;&quot;&quot;)</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N70 M30</td>
<td></td>
</tr>
<tr>
<td>N55 M3 S500 G90 G0 X0 Z0</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N87 MSG (&quot;tapping&quot;)</td>
<td>; Deactivate fast retraction before tapping.</td>
</tr>
<tr>
<td>N88 LFOP</td>
<td></td>
</tr>
<tr>
<td>N89 CYCLE...</td>
<td>; Tapping cycle with G33.</td>
</tr>
<tr>
<td>N90 MSG (&quot;&quot;&quot;)</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N99 M30</td>
<td></td>
</tr>
</tbody>
</table>

Behavior at power on and reset

After power on and reset, the following settings are activated:

- Initial settings for the retraction motion (LFON /LFOF) and retraction direction (LFTXT/LFWP): MD20150 $MC_GCODE_RESET_VALUES
- Retraction path: MD21200 $MC_LIFTFAST_DIST
17.2.3.5 Convex thread (G335, G336)

Function

The G functions G335 and G336 can be used to turn convex threads (= differing to the cylindrical form). Application is the machining of extremely large components that sag in the machine because of their self-weight. Paraxial thread would result in the thread being too small in the middle of the component. This can be compensated with convex threads.

![Figure 17-2 Turning a convex thread](image)

Programming

The turning of a convex thread is programmed with G335 or G336:

- **G335**: Turning of a convex thread on a circular tool path in a **clockwise direction**
- **G336**: Turning of a convex thread on a circular tool path in a **counter-clockwise direction**

The programming is performed first as for a linear thread by specifying the axial block end points and the pitch via parameters I, J and K. An arc is also specified. As for G2/G3, this can be programmed via the center point, radius, opening angle or intermediate point specification. When programming the convex thread with center point programming, the following must be taken into account: Since I, J and K are used for the pitch in thread cutting, the circle parameters in the center point programming must be programmed with IR=..., JR=... and KR=....

| IR=... | Cartesian coordinate for the circle center point in the X direction |
| JR=... | Cartesian coordinate for the circle center point in the Y direction |
| KR=... | Cartesian coordinate for the circle center point in the Z direction |

**Note**

IR, JR and KR are the default values of the interpolation parameter names for a convex thread that can be set via machine data MD10651 $MN_IPO_PARAM_THREAD_NAME_TAB.

Optionally, a starting point offset SF can also be specified.
Syntax

The syntax for the programming of a convex thread therefore has the following general form:

G335/G336 <axis target point coordinate(s)> <pitch> <arc> [<<starting point offset>]

Permissible arc areas

The arc programmed at G335/G336 must be in an area in which the specified thread main axis (I, J or K) has the main axis share on the arc over the entire arc:

![Permissible areas for the Z axis](image1)

Permissible areas for the Z axis (pitch programmed with K)

A change of the thread main axis as shown in the following figure is not permitted:

![A change of the thread main axis](image2)

Boundary conditions

Frames

G335 and G336 are also possible with active frames. However, you must ensure that the permissible arc areas are maintained in the basic coordinate system (BCS).
Response for ...

The G335/G336 response for:

- Power On / Power Off
- Mode change
- NCK / mode group / channel / part program end reset
- Block search / REPOS / ASUB
- Alarms / emergency stop / malfunctions

corresponds to the behavior for G33/G34/G35.

There are no specific restrictions.

Examples

Example 1: Convex thread in the clockwise direction with end and center point programming

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N5 G0 G18 X50 Z50</td>
<td>; Approach starting point.</td>
</tr>
<tr>
<td>N10 G335 Z100 K=3.5 KR=25 IR=-20 SF=90</td>
<td>; Turn convex thread in the clockwise direction.</td>
</tr>
</tbody>
</table>

![Figure 17-4 Convex thread in the clockwise direction with end and center point programming](image)

Example 2: Convex thread in the counter-clockwise direction with end and center point programming

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N5 G0 G18 X50 Z50</td>
<td>; Approach starting point.</td>
</tr>
<tr>
<td>N10 G336 Z100 K=3.5 KR=25 IR=20 SF=90</td>
<td>; Turn convex thread in the counter-clockwise direction.</td>
</tr>
</tbody>
</table>
Example 3: Convex thread in the clockwise direction with end point and radius programming

Program code

N5 G0 G18 X50 Z50
N10 G335 Z100 K=3.5 CR=32 SF=90

Example 4: Convex thread in the clockwise direction with end point and opening angle programming

Program code

N5 G0 G18 X50 Z50
N10 G335 Z100 K=3.5 AR=102.75 SF=90
**Example 5: Convex thread in the clockwise direction with center point and opening angle programming**

**Program code**

```
N5 G0 G18 X50 Z50
N10 G335 K=3.5 KR=25 IR=-20 AR=102.75 SF=90
```

**Example 6: Convex thread in the clockwise direction with end and intermediate point programming**

**Program code**

```
N5 G0 G18 X50 Z50
N10 G335 Z100 K=-3.5 I1=60 K1=64
```
17.2.4 Feedrate for tapping without compensating chuck (G331, G332)

Function

A thread can be tapped without compensating chuck with the functions G331 (tapping) and G332 (tapping retraction).

Requirement

The technical requirement for tapping without compensating chuck is a position-controlled spindle with position measuring system.

Speed S, feedrate F, thread pitch

A revolutional feedrate [mm/rev] is used for G331 and G332. The revolutional feedrate is defined by programming the thread pitch [mm/rev].

The speed of the axes for the thread length is calculated from the programmed spindle speed S and the thread pitch.

Feedrate F [mm/min] = speed S [rev/min] * pitch [mm/rev]

Override

The revolutional feedrate in G331 and G332 can be influenced by an override.

Depending on the configuration, the override affects either the spindle speed or the path feedrate:
MD12090 $MN_OVR_FUNCTION_MASK

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>The override influences the spindle speed (initial setting). Depending on the setting in the machine data: MD12080 $MN_OVR_REFERENCE_IS_PROG_FEED the override is related either to the programmed spindle speed or to the configured spindle speed limitation.</td>
</tr>
<tr>
<td>1</td>
<td>The override influences the path feedrate Depending on the setting in machine data: MD12082 $MN_OVR_REFERENCE_IS_MIN_FEED the override is related either to the programmed path feedrate or to the configured path feedrate limitation.</td>
<td></td>
</tr>
</tbody>
</table>

Note
The following overrides are not effective in G331 and G332:
- Programmable path feedrate override OVR
- Rapid traverse override

Inhibiting stop events for G331/G332
During tapping, a stop can be prevented if the block contains a path motion or a G4 as follows:
MD11550 $MN_STOP_MODE_MASK, bit 0 = 0
The stop which was activated previously is possible again after G331/G332 has been executed.

References
For further information on G331/G332, see Programming Manual Fundamentals.

17.2.5 Feedrate for tapping with compensating chuck (G63)

Function
G63 is a subfunction for tapping threads using a tap with compensating chuck. An encoder (position encoder) is not required.

Speed S, feedrate F, thread pitch
With G63, a speed S must be programmed for the spindle and a feedrate F for the infeed axis (axis for thread length).
The feedrate F must be calculated by the programmer on the basis of the speed S and the thread pitch.
Feedrate F [mm/min] = speed S [rev/min] * pitch [mm/rev]
17.3 Feedrate for positioning axes (FA)

Function

The velocity of a positioning axis is programmed with axis-specific feedrate FA. FA is modal. The feedrate is always G94.

Note

The maximum axis velocity (MD32000 $MA_MAX_AX_VELO) is not exceeded.

Programming

No more than five axis-specific feedrates can be programmed in each part program block.

Syntax:

FA[<positioning axis>] = <feedrate value>

<positioning axis>: Name of the channel axis
(MD20080 $MC_AXCONF_CHANAX_NAME_TAB)

<feedrate value>: Feedrate
Value range: 0.001…999 999.999 mm/min, deg/min or 0.001…39 999.999 inch/min

Default setting

If no axial feedrate FA is programmed, the axial default setting is applied:
MD32060 $MA_POS_AX_VELO (initial setting for positioning axis velocity)

Output to PLC

The feedrate value can be output to the PLC:
- To the channel-specific NC/PLC interface via:
  DB21, ... DBB158 - DBB193
- To the axis-specific NC/PLC interface via:
  DB31, ... DBB78 - DBB81

The output time is specified with the machine data:

References

For more information on G63, see Programming Manual Fundamentals.
MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of F functions)
The output is suppressed in the default setting (MD22240 = 3), because drops in velocity can occur through the output of F functions to the NC/PLC interface in continuous-path mode. For more information, see Description of Functions "H2: Auxiliary function outputs to PLC (Page 377)".

Reset behavior

The behavior after the end of program or NC reset is specified by the machine data:

MD22410 $MC_F_VALUES_ACTIVE_AFTER_RESET (F function is active even after reset)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The default values are effective after NC reset.</td>
</tr>
<tr>
<td>1</td>
<td>The last programmed FA values are effective after NC reset.</td>
</tr>
</tbody>
</table>

17.4 Feedrate control

17.4.1 Feedrate disable and feedrate/spindle stop

Function

The "Feed disable" or "Feed/spindle stop" brings the axes to a standstill with adherence to the braking characteristics and the path contour (exception: G33 block).

Channel-specific feedrate disable

The NC/PLC interface signal:
DB21, ... DBX6.0 (feedrate disable) stops all axes (geometry and auxiliary axes) of a channel in all modes.

<table>
<thead>
<tr>
<th>Effectiveness of the channel-specific feedrate disable:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• With active G33, G34, G35: Not effective</td>
</tr>
<tr>
<td>• With active G63: Effective</td>
</tr>
<tr>
<td>• With active G331, G332: Effective</td>
</tr>
</tbody>
</table>

"Feed stop" for geometry axes in JOG mode

The NC/PLC interface signals:
DB21, ... DBX12.3 (feed stop for geometry axis 1)
DB21, ... DBX16.3 (feed stop for geometry axis 2)
DB21, ... DBX20.3 (feed stop for geometry axis 3)
stop the relevant geometry axes if a channel in JOG mode.
**Axis-specific "Feed stop"**

The axis-specific NC/PLC interface signal:
DB31, ... DBX4.3 (feed stop)
stops the respective machine axis.

In automatic mode:
- If the "Feed stop" is performed for a path axis, all the axes traversed in the current block and all axes participating in the axis group are stopped.
- If the "Feed stop" is performed for a positioning axis, only this axis is stopped.

Only the current axis is stopped in JOG mode.

<table>
<thead>
<tr>
<th>Effectiveness of the axis-specific &quot;Feed stop&quot;:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• With active G33, G34, G35: Effective (causes contour deviations)</td>
</tr>
<tr>
<td>• With active G63: Effective</td>
</tr>
<tr>
<td>• With active G331, G332: Effective</td>
</tr>
</tbody>
</table>

**Axis/spindle disable**

With active "Axis/spindle disable":
DB31, ... DBX1.3 = 1

The axial PLC interlocks "No controller enable" and "Feed stop" have no effect.

However, the axial and channel-specific override are effective.

**"Spindle stop"**

The NC/PLC interface signal:
DB31, ... DBX4.3 (spindle stop)
stops the respective spindle.

<table>
<thead>
<tr>
<th>Effectiveness of the &quot;Spindle stop&quot; function</th>
</tr>
</thead>
<tbody>
<tr>
<td>• With active G33, G34, G35: Effective (may cause contour deviations depending on dynamic characteristics)</td>
</tr>
<tr>
<td>• With active G63: Effective</td>
</tr>
<tr>
<td>• With active G331, G332: Not effective</td>
</tr>
</tbody>
</table>

**17.4.2 Feedrate override on machine control panel**

**Function**

With the "Feedrate override via the machine control panel", the user can locally increase or decrease the path feedrate at the machine as a percentage with immediate effect. To achieve this, the programmed feedrates are multiplied with the override values available at the NC/PLC interface.

The feedrate can be changed axis-specifically for positioning axes.
The "Spindle override" can be used to change the spindle speed and the cutting rate (G96, G961).

With a feedrate change, the axial acceleration and velocity limits are maintained. There are no contour errors along the path.

The feedrate override can be changed separately for path and position axes.

The overrides influence the programmed values or the limits (e.g. G26, LIMS for spindle speed).

**Channel-specific feedrate and rapid traverse**

For feedrate and rapid traverse override, dedicated enable signals and correction/offset factors are available in the NC/PLC interface:

DB21, ... DBX6.7 (feedrate override active)
DB21, ... DBB4 (feedrate override)
DB21, ... DBX6.6 (rapid traverse override active)
DB21, ... DBB5 (rapid traverse override)

The override factors can be specified from the PLC either in the binary or gray-coded format. The format is communicated to the NC via the following machine data:

MD12020 $MN_OVR_FEED_IS_GRAY_CODE (path feedrate override switch gray-coded)
MD12040 $MN_OVR_RAPID_IS_GRAY_CODE (rapid traverse override switch gray-coded)

The following permanent assignment applies to binary code:

<table>
<thead>
<tr>
<th>Binary code</th>
<th>decimal</th>
<th>Override factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0</td>
<td>0.00 ± 0%</td>
</tr>
<tr>
<td>00000001</td>
<td>1</td>
<td>0.01 ± 1%</td>
</tr>
<tr>
<td>00000010</td>
<td>2</td>
<td>0.02 ± 2%</td>
</tr>
<tr>
<td>00000011</td>
<td>3</td>
<td>0.03 ± 3%</td>
</tr>
<tr>
<td>00000100</td>
<td>4</td>
<td>0.04 ± 4%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>01100100</td>
<td>100</td>
<td>1.00 ± 100%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>11001000</td>
<td>200</td>
<td>2.00 ± 200%</td>
</tr>
</tbody>
</table>

With Gray coding, the override factors corresponding to the switch position must be entered in the following machine data:

MD12030 $MN_OVR_FACTOR_FEEDRATE [<n>] (evaluation of the path feedrate override switch)
MD12050 $MN_OVR_FACTOR_RAPID_TRA [<n>] (evaluation of the rapid traverse override switch)

An active feedrate override has an effect on all path axes that are assigned to the current channel. An active rapid traverse override has an effect on all the axes that are traversed with rapid traverse and that are assigned to the current channel.
No rapid traverse override switch available

If there is no dedicated rapid traverse override switch, you can choose between rapid traverse override and feedrate override. The override to be active can be selected via the PLC or operator panel front. When rapid traverse override is active, the feedrate override values are limited to 100%.

- When the rapid traverse override is activated via the operator panel front, the basic PLC program:
  - Transfers the selection of the feedrate override for rapid traverse on the activation signal for the rapid traverse override:
    \[ DB21, ..., DBX6.6 = DB21, ..., DBX25.3 \]
  - Transfers the feedrate override value in the rapid traverse override value:
    \[ DB21, ..., DBB5 = DB21, ..., DBB4 \]

- When the rapid traverse override is selected via the PLC, the PLC user program:
  - Must set the activation signal for the rapid traverse override:
    \[ DB21, ..., DBX6.6 = 1 \]
  - Must transfer the feedrate override value in the rapid traverse override value:
    \[ DB21, ..., DBB5 = DB21, ..., DBB4 \]

<table>
<thead>
<tr>
<th>Effectiveness of the channel-specific feedrate and rapid traverse override:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• With active G33, G34, G35: Not effective</td>
</tr>
<tr>
<td>• With active G63: Not effective</td>
</tr>
<tr>
<td>• With active G331, G332: Not effective</td>
</tr>
</tbody>
</table>

Reference velocity for path feedrate override

The reference velocity for the "Path feedrate override via machine control panel" can be set differently to the standard feedrate (= programmed feedrate).

\[ MD12082 $MN_OVR_REFERENCE_IS_MIN_FEED \]

Axis-specific feedrate override

An enable signal and a byte for the feedrate override factor are in the NC/PLC interface for each positioning axis.

\[ DB31, ..., DBX1.7 (override effective) \]

\[ DB31, ..., DBB0 (feedrate override) \]

The override factor can be specified from the PLC either in the binary or gray-coded format. The format is communicated to the NC via the following machine data:

\[ MD12000 $MN_OVR_AX_IS_GRAY_CODE (axis feedrate override switch gray-coded) \]

The following permanent assignment applies to binary code:

<table>
<thead>
<tr>
<th>Binary code</th>
<th>Decimal</th>
<th>Override factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0</td>
<td>0.00 ± 0%</td>
</tr>
<tr>
<td>00000001</td>
<td>1</td>
<td>0.01 ± 1%</td>
</tr>
</tbody>
</table>
With Gray coding, the override factors corresponding to the switch position must be entered in the following machine data:

MD12010 $MN_OVR_FACTOR_AX_SPEED [<n>] (evaluation of the axis feedrate override switch)

Effectiveness of the axis-specific feedrate override:

- With active G33, G34, G35: Not effective
- With active G63: Not effective (the override is set in the NC permanently to 100%)
- With active G331, G332: Not effective (the override is set in the NC permanently to 100%)

Spindle override

One enable signal and one byte for the spindle override factor are available in the NC/PLC interface for each spindle.

DB31, ... DBX1.7 (override effective)

DB31, ... DBB19 (spindle override)

The override factor can be specified from the PLC either in the binary or gray-coded format. The format is communicated to the NC via the following machine data:

MD12060 $MN_OVR_SPIND_IS_GRAY_CODE (spindle override switch gray-coded)

The following permanent assignment applies to binary code:

<table>
<thead>
<tr>
<th>Binary code</th>
<th>decimal</th>
<th>Override factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0</td>
<td>0.00 ± 0%</td>
</tr>
<tr>
<td>00000001</td>
<td>1</td>
<td>0.01 ± 1%</td>
</tr>
<tr>
<td>00000010</td>
<td>2</td>
<td>0.02 ± 2%</td>
</tr>
<tr>
<td>00000011</td>
<td>3</td>
<td>0.03 ± 3%</td>
</tr>
<tr>
<td>00000100</td>
<td>4</td>
<td>0.04 ± 4%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>01100100</td>
<td>100</td>
<td>1.00 ± 100%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>11001000</td>
<td>200</td>
<td>2.00 ± 200%</td>
</tr>
</tbody>
</table>

With Gray coding, the override factors corresponding to the switch position must be entered in the following machine data:
MD12070 $MN_OVR_FACTOR_SPIND_SPEED [<n>] (evaluation of the spindle override switch)

<table>
<thead>
<tr>
<th>Effectiveness of the &quot;Spindle override&quot;:</th>
</tr>
</thead>
<tbody>
<tr>
<td>With active G33, G34, G35:</td>
</tr>
<tr>
<td>With active G63:</td>
</tr>
<tr>
<td>With active G331, G332:</td>
</tr>
</tbody>
</table>

Reference to spindle override

The spindle override can refer to the speed or the programmed speed limited by the machine or setting data. The setting is realized via:

MD12080 $MN_OVR_REFERENCE_IS_PROG_FEED (override reference velocity)

Limiting the override factor

For binary-coded override factors, the maximum possible overrides for path feedrate, axis feedrate and spindle speed can be limited:

MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary coded override switch)

Override active

For overrides that have been enabled, the specified override values entered via the machine control panel become immediately active in all operating modes and machine functions.

Override inactive

An override factor of 100% is internally effective if an override is not activated. The override factor at the NC/PLC interface is not evaluated.

An exception is the zero setting for a binary interface and the 1st switch setting for a gray-coded interface. In these cases, the override factors entered at the NC/PLC interface are evaluated. For a binary interface, the override factor is always 0%. For a gray-coded interface, the value entered in machine data for the 1st switch position value is output as override value. It should be populated with 0.

17.4.3 Programmable feedrate override

Function

The "Programmable feedrate override" function can be used to change the velocity level of path and positioning axes via the part program.
Programming

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OVR=&lt;value&gt;</td>
<td>Feedrate change for path feedrate F</td>
</tr>
<tr>
<td>OVRA[&lt;axis&gt;=&lt;value&gt;]</td>
<td>Feedrate change for positioning feedrate FA</td>
</tr>
</tbody>
</table>

The programmable range is between 0 and 200%.
Default setting: 100%

Effectiveness

The NC/PLC interface signals DB21, ..., DBB6 (rapid traverse or feedrate override active) and DB31, ..., DBX1.7 (axis-specific override active) do not refer to the programmable feedrate override. The programmable feedrate override remains active when these signals are deactivated.

The effective override is calculated from the product of the "Programmable feedrate override" and the "Feedrate override on machine control panel (Page 1335)".

The default setting for the "Programmable feedrate override" is 100%.

The default setting is effective:
- If no feedrate override is programmed or
- After reset if the machine data:
  MD22410 $MC_F_VALUES_ACTIVE_AFTER_RESET (F function is active even after reset)
  is not set.

Note

OVR is not effective with G33, G34, G35.

17.4.4 Dry run feedrate

Function

The dry run feedrate is used when testing part programs without machining the workpiece in order to allow the program or program sections to execute with an increased path feedrate, for example.

Activation

The dry run feedrate can be selected in the automatic modes and activated from the PLC or the operator panel front.

When activated from the operator panel front, the interface signal:
DB21, ..., DBX24.6 (dry run feedrate selected)
is set and transferred from the basic PLC program to the interface signal:
DB21, ... DBX0.6 (activate dry run feedrate).

When selected on the PLC, the interface signal DB21, ... DBX0.6 (activate dry run feedrate) must be set from the PLC user program.

**Effectiveness**

As long as the “Activate dry run feedrate” interface signal is set, instead of the programmed feedrate, the feedrate value set via SD42100 DRY_RUN_FEED is effective in the way specified via SD42101 $SC_DRY_RUN_FEED_MODE (see parameterization):

The dry run feedrate is always interpreted as linear feedrate (G94).

**Parameterization**

**Activation of dry run feedrate**

The time of activation depends on the setting in the machine data:

MD10704 $MN_DRYRUN_MASK (activation of dry run feedrate)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The dry run feedrate may only be switched on and off at the end of the block (default setting).</td>
</tr>
</tbody>
</table>
| 1     | The dry run feedrate can also be activated during the program processing (in the part program block).  
**Notice:**  
Activation during processing triggers an internal reorganization operation on the controller which causes the axes to be stopped for a short time. This can affect the surface finish of the workpiece being machined. |
| 2     | The dry run feedrate can be activated/deactivated at any time without the axes being stopped. The function only takes effect with a block "later" in the program run. |

**Changing the dry run feedrate**

The feedrate for the dry run is entered in the setting data:

SD42100 $SC_DRY_RUN_FEED (dry run feedrate)

The setting data can be changed via the operator panel front in the "Parameters" operating area.

If the selection has been accepted by the NCK, the following NC/PLC interface signal is set:

DB21, ... DBX318.6 (dry run feedrate active)

"DRY" is displayed in the operator panel front status bar to indicate an active dry run feedrate if:

- Selection took place during the program stop at the end of a block or
- The machine data MD10704 $MN_DRYRUN_MASK was set to "1" during the program execution
**Mode of operation of the dry run feedrate**

The mode of operation of the dry run feedrate entered in SD42100 can be set via the setting data:

SD42101 $SC_DRY_RUN_FEED_MODE

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The programmed feedrate is compared to the dry run feedrate in SD42100 and then traversing is performed with the higher of the two feedrates (default setting).</td>
</tr>
<tr>
<td>1</td>
<td>The programmed feedrate is compared to the dry run feedrate in SD42100 and then traversing is performed with the lower of the two feedrates.</td>
</tr>
<tr>
<td>2</td>
<td>The dry run feedrate entered in SD42100 takes effect directly, irrespective of the programmed velocity.</td>
</tr>
<tr>
<td>3-9</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>As for configuration 0, except for thread cutting (G33, G34, G35) and tapping (G331, G332, G63). These functions are executed as programmed.</td>
</tr>
<tr>
<td>11</td>
<td>As for configuration 1, except for thread cutting (G33, G34, G35) and tapping (G331, G332, G63). These functions are executed as programmed.</td>
</tr>
<tr>
<td>12</td>
<td>As for configuration 2, except for thread cutting (G33, G34, G35) and tapping (G331, G332, G63). These functions are executed as programmed.</td>
</tr>
</tbody>
</table>

**17.4.5 Multiple feedrate values in one block**

**Function**

The function "Multiple feedrate values in one block" can be used to activate six different feedrate values of an NC block, a dwell time or a retraction motion-synchronously, depending on the external digital and/or analog inputs.

When the input for the sparking out time or retraction path is activated, the distance-to-go for the path axes or the particular single axis is deleted and the dwell time or retraction is started.

The retraction is started within an IPO cycle.

**Signals**

The input signals are combined in one input byte for the function. A fixed functional assignment applies within the byte.

| Table 17-1 Input byte for the "Multiple feedrates in one block" function |
|--------------------------|---|---|---|---|---|---|---|---|
| Bit  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Input no. | I7 | I6 | I5 | I4 | I3 | I2 | I1 | I0 |
| Feedrate address | F7 | F6 | F5 | F4 | F3 | F2 | ST | SR |

I7 to I2: Activation of feedrates F7 to F2
E1: Activation of the dwell time ST/STA (in seconds)
I0: Activation of the retraction motion SR/SRA
Priority of the signals

The signals are scanned in ascending order starting at I0. Therefore, the retraction motion (SR) has the highest priority and the feedrate F7 the lowest priority.

SR and ST end the feedrate motions that were activated with F2 to F7.
SR also ends ST, i.e. the complete function.

The signal with the highest priority determines the current feedrate.

The response to loss of the respective highest-priority input (F2 - F7) can be defined with the machine data:

MD21230 $MC_MULTFEED_STORE_MASK (storage behavior for the "Multiple feedrate values in one block" function)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 ... 7</td>
<td>0</td>
<td>With the loss of the respective highest-priority input, the associated feedrate is not retained (default setting).</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Set bit 2 to 7 ensures that the associated feedrate (F2 to F7) that was selected by the respective highest-priority input signal is also retained when there is a loss of the input signal and a lower-priority input is active.</td>
</tr>
</tbody>
</table>

The end-of-block criterion is satisfied when:
- The programmed end position is reached
- The retraction motion ends (SR)
- The dwell time elapses (ST)

Hardware assignment

The input byte for the "Multiple feedrate values in one block" function can be assigned a maximum of two digital input bytes or comparator input bytes of the NCK I/O:

MD21220 $MC_MULTFEED_ASSIGN_FASTIN (assignment of the input bytes of the NCK I/O for "Multiple feedrate values in one block"), bit 0 ... 15

The input bits can also be inverted:

MD21220 $MC_MULTFEED_ASSIGN_FASTIN, bit 16 ... 31
17.4 Feedrate control

The assignment of the digital input bytes and parameterization of the comparators are described in:

References:
Function Manual, Extended Functions; Digital and Analog NCK I/O (A4)

Programming

Path motion
The path feedrate is programmed under the address $F$ and remains valid until an input signal is present. This value acts modally.

$F2=\ldots$ to $F7=\ldots$ can be used in addition to the path feedrate to program up to six further feedrates in the block. The numerical expansion indicates the bit number of the input that activates the feedrate when changed:

Example:

$F7=1000$ ;7 corresponds to input bit 7

The programmed values act non-modally. The path feedrate programmed under $F$ applies in the next block.
Dwell (sparking out time) and retraction path are programmed under separate addresses in the block:

ST=...  
Dwell time (for grinding sparking out time)
SR=...  
Retraction path

These addresses apply non-modally.

**Axial motion**

The axial feedrates are programmed under address FA and remain valid until an input signal is present. They act modally.

FMA[2,<axis>]=... to FMA[7,<axis>]=... can be used to program up to six further feedrates per axis in the block.

The first expression in square brackets indicates the bit number of the input that activates the feedrate when changed. The second expression indicates the axis to which the feedrate applies.

Example:

FMA[3,Y]=1000 ; Axial feedrate for Y axis, corresponds to input bit 3

The values programmed under FMA act non-modally. The feedrate programmed under FA applies to the next block.

Dwell (sparking out time) and retraction path can also be defined for a single axis:

STA[<axis>]=...  
Axial dwell time (sparking out time)
SRA[<axis>]=...  
Axial retraction path

The expression in square brackets indicates the axis for which the sparking out time and retraction path apply.

Examples:

STA[X]=2.5 ; The sparking out time for the X axis is 2.5 seconds.
SRA[X]=3.5 ; The retraction path for the X axis is 3.5 (unit e.g. mm).

These addresses apply non-modally.

---

**Note**

**Retraction path**

The unit for the retraction path refers to the current valid unit of measurement (mm or inch).

The reverse stroke is always made in the opposite direction to the current motion. SR/SRA always programs the value for the reverse stroke. No sign is programmed.
17.4 Feedrate control

Note
POS instead of POSA
If feedrates, sparking out time (dwell time) or return path are programmed for an axis on account of an external input, this axis must not be programmed as POSA axis (positioning axis over multiple blocks) in this block.

Note
Status query
It is also possible to poll the status of an input for synchronous commands of various axes.

Note
LookAhead
Look Ahead is also active for multiple feedrates in one block. In this way, the current feedrate can be restricted by the Look Ahead value.

Application
The "Multiple feedrate values in one block" function is used primarily for grinding, but is not restricted to it.

Typical applications are, for example:

- Analog or digital calipers
  Depending on whether the external inputs are analog or digital, various feedrate values, a dwell time and a retraction path can be activated. The limit values are defined via the setting data.

- Switching from infeed to working feedrate via proximity switch
**Example**

Internal grinding of a conical ring, where the actual diameter is determined using calipers and, depending on the limits, the feedrate value required for roughing, finishing or fine finishing is activated. The position of the calipers also provides the end position. Thus, the block end criterion is determined not only by the programmed axis position of the infeed axis but also by the calipers.

![Diagram](image)

**17.4.6 Fixed feedrate values**

**Function**

The "Fixed feedrate values" function can be used to activate fixed feedrates (max. four) defined via the machine data instead of the programmed feedrate or the configured JOG velocities. The function is available in AUTOMATIC and JOG mode.

**Behavior in AUTOMATIC mode**

The contour travels at the activated fixed feedrate, instead of using the programmed feedrate.
Behavior in JOG mode
The axis is traversed with the activated fixed feedrate instead of the configured JOG velocity / JOG rapid traverse velocity. The travel direction is specified via the interface signal.

Parameterization
The setting of the fixed feedrates is performed:

- For linear axes with the machine data:
  MD12202 $MN_PERMANENT_FEED\langle n\rangle$
- For rotary axes with the machine data:
  MD12204 $MN_PERMANENT_ROT_AX_FEED\langle n\rangle$

where \(\langle n\rangle = 0, 1, 2, 3\) (for fixed feedrate 1, 2, 3, 4)

Note
The fixed feedrates are always linear feedrate values. Switchover to linear feedrate is conducted internally even in case of revolutional feedrate.

Activation
The fixed feedrates are activated via NC/PLC interface signals:

- In AUTOMATIC mode for path/geometry axes using the channel-specific interface signals:
  DB21, ..., DBX29.0 (activate fixed feedrate 1)
  DB21, ..., DBX29.1 (activate fixed feedrate 2)
  DB21, ..., DBX29.2 (activate fixed feedrate 3)
  DB21, ..., DBX29.3 (activate fixed feedrate 4)
- In JOG mode for machine axes using the axis-specific interface signals:
  DB31, ..., DBX3.2 (activate fixed feedrate 1)
  DB31, ..., DBX3.3 (activate fixed feedrate 2)
  DB31, ..., DBX3.4 (activate fixed feedrate 3)
  DB31, ..., DBX3.5 (activate fixed feedrate 4)

Supplementary conditions

Effectiveness
The function "Fixed feedrate values" is **not** active:

- For spindles
- For positioning axes
- When tapping

Override = 0
The traversing behavior for override = 0 depends on the setting in machine data:

MD12200 $MN_RUN_OVERRIDE_0
DRF offset

The DRF offset cannot be activated for a selected fixed feedrate.

17.4.7 Programmable feedrate characteristics

Function

To permit flexible definition of the feedrate characteristic, the feedrate programming according to DIN 66025 has been extended by linear and cubic characteristics.

The cubic profiles can be programmed directly or as an interpolating spline.

Programming

You can program the following feedrate profiles:

- FNORM
  Behavior in accordance with DIN 66025 (default setting).
  An F-value programmed in the block is applied over the entire path of the block, and is subsequently regarded as a fixed modal value.

- FLIN
  An F value programmed in the block is traversed linearly over the path from the current value at the beginning of the block to the end of the block, and is subsequently regarded as modal value.

- FCUB
  The non-modal programmed F values (relative to the end of the block) are connected by a spline. The spline starts and ends tangentially to the previous or following feedrate setting. If the F address is missing in one block, then the last programmed F value is used.

- FPO
  The F address [syntax: F=FPO(...,...,...)] designates the characteristic of the feedrate via a polynomial from the current value to the end of the block in which it was programmed. The end value is treated as modal from there onwards.

Parameterization

If FLIN and FCUB are used in connection with compression COMPON, a tolerance can be defined for the path feedrate:

MD20172 $MC_COMPRESS_VELO_TOL (max. permissible deviation of the path feedrate with compression)

Supplementary conditions

FLIN/FCUB

The path velocity profile programmed with FLIN or FCUB is not active together with revolutionary feedrate for G95 as well as with constant cutting rate with G96/G961 and G97/G971.
References
For further information on the programmable feedrate characteristics, see Programming Manual, Job Planning.

17.4.8 Feedrate for chamfer/rounding FRC, FRCM

The machining conditions can change significantly during surface transitions to chamfer/rounding. Hence, the chamfer/rounding contour elements require dedicated, optimized feedrate values to achieve the desired surface quality.

Function

The feedrate for chamfer/rounding can be programmed via NC addresses.

Programming

Syntax:

... \text{ FRC/FRCM=}<\text{value}> \\

Meaning:

<table>
<thead>
<tr>
<th>FRC:</th>
<th>Non-modal feedrate for chamfer/rounding</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRCM:</td>
<td>Modal feedrate for chamfer/rounding</td>
</tr>
<tr>
<td>&lt;\text{value}&gt;:</td>
<td>The feedrate is interpreted according to the active feedrate type:</td>
</tr>
<tr>
<td></td>
<td>● G94, G961, G971: Feedrate in mm/min or inch/min or (^\circ/min)</td>
</tr>
<tr>
<td></td>
<td>● G95, G96, G97: Revolutonal feedrate in mm/rev or inch/rev</td>
</tr>
</tbody>
</table>

Note

FRC is only effective if a chamfer/rounding is programmed in the block or if RNDM has been activated.

FRC overwrites the F or FRCM value in the current block.

The feedrate programmed under FRC must be greater than zero.

FRCM=0 activates the feedrate programmed under F for chamfering/rounding.

Parameterization

Assignment of the chamfer/rounding to the previous or following block

The feedrate type (G94, G95, G96, G961 ...) and therefore the conversion to the internal format must be consistent within the block for F and FRC/FRCM. In this context, the following machine data must be taken into account:
MD20201 $MC_CHFRND_MODE_MASK (chamfer/rounding behavior)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>The technology of the chamfer/rounding (feedrate, feedrate type, M commands, etc.) is determined by the following block (default setting).</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>The technology of the chamfer/rounding is determined by the previous block (recommended setting).</td>
</tr>
</tbody>
</table>

**Maximum number of empty blocks**

The number of blocks without traversing information in the compensation plane (empty blocks) permitted between two blocks with traversing information during active chamfer/rounding, is limited. The maximum number is specified in the machine data:

MD20200 $MC_CHFRND_MAXNUM_DUMMY_BLOCKS (empty blocks for chamfer/radii)

**Supplementary conditions**

**FLIN/FCUB**

Feedrate interpolation FLIN and FCUB is not possible for chamfer/rounding.

**G0**

FRC/FRCM is not active when a chamfer is traversed with G0. The programming is possible in accordance with the F value without error message.

**Change G94 ↔ G95**

If FRCM is programmed, the FRCM value will need to be reprogrammed like F on change G94 ↔ G95, etc. If only F is reprogrammed and if the feedrate type FRCM > 0 before the change, an error message will be output.

**Example**

**Example 1: MD20201 bit 0 = 0; take feedrate from following block (default setting!)**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G0 X0 Y0 G17 F100 G94</td>
<td></td>
</tr>
<tr>
<td>N20 G1 X10 CHF=2 ; Chamfer N20-N30 with F=100 mm/min</td>
<td></td>
</tr>
<tr>
<td>N30 Y10 CHF=4 ; Chamfer N30-N40 with FRC=200 mm/min</td>
<td></td>
</tr>
<tr>
<td>N40 X20 CHF=3 FRC=200 ; Chamfer N40-N60 with FRCM=50 mm/min</td>
<td></td>
</tr>
<tr>
<td>N50 RNDM=2 FRCM=50</td>
<td></td>
</tr>
<tr>
<td>N60 Y20 ; Modal rounding N60-N70 with FRCM=50 mm/min</td>
<td></td>
</tr>
<tr>
<td>N70 X30 ; Modal rounding N70-N80 with FRCM=50 mm/min</td>
<td></td>
</tr>
<tr>
<td>N80 Y30 CHF=3 FRC=100 ; Chamfer N80-N90 with FRC=100 mm/min</td>
<td></td>
</tr>
<tr>
<td>N90 X40 ; Modal rounding N90-N100 with F=100 mm/min (delection of FRCM)</td>
<td></td>
</tr>
<tr>
<td>N100 Y40 FRCM=0 ; Modal rounding N100-N120 with G95 FRC=1 mm/rev</td>
<td></td>
</tr>
<tr>
<td>N110 S1000 M3</td>
<td></td>
</tr>
</tbody>
</table>
### 17.4 Feedrate control

#### Example 2: MD20201 bit 0 = 1; take feedrate from previous block (recommended setting!)

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G0 X0 Y0 G17 F100 G94</td>
<td></td>
</tr>
<tr>
<td>N20 G1 X10 CHF=2</td>
<td>; Chamfer N20-N30 with F=100 mm/min</td>
</tr>
<tr>
<td>N30 Y10 CHF=4 FRC=120</td>
<td>; Chamfer N30-N40 with FRC=120 mm/min</td>
</tr>
<tr>
<td>N40 X20 CHF=3 FRC=200</td>
<td>; Chamfer N40-N60 with FRC=200 mm/min</td>
</tr>
<tr>
<td>N50 RNDM=2 FRCM=50</td>
<td></td>
</tr>
<tr>
<td>N60 Y20</td>
<td>; Modal rounding N60-N70 with FRCM=50 mm/min</td>
</tr>
<tr>
<td>N70 X30</td>
<td>; Modal rounding N70-N80 with FRCM=50 mm/min</td>
</tr>
<tr>
<td>N80 Y30 CHF=3 FRC=100</td>
<td>; Chamfer N80-N90 with FRC=100 mm/min</td>
</tr>
<tr>
<td>N90 X40</td>
<td>; Modal rounding N90-N100 with FRCM=50 mm/min</td>
</tr>
<tr>
<td>N100 Y40 FRCM=0</td>
<td>; Modal rounding N100-N120 with F=100 mm/min</td>
</tr>
<tr>
<td>N110 S1000 M3</td>
<td></td>
</tr>
<tr>
<td>N120 X50 CHF=4 G95 F3 FRC=1</td>
<td>; Chamfer N120-N130 with G95 FRC=1 mm/rev</td>
</tr>
<tr>
<td>N130 Y50</td>
<td>; Modal rounding N130-N140 with F=3 mm/rev</td>
</tr>
<tr>
<td>N140 X60</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 17.4.9 Non-modal feedrate FB

**Function**

The "Non-modal feedrate" function can be used to define a separate feedrate for a single part program block. After this block, the previous modal path feedrate is active again.

**Programming**

**Syntax:**

```
... FB=<value>
```

**Meaning:**

- **FB:** Separate feedrate for the current block
- **<value>:** The feedrate is interpreted according to the active feedrate type:
  - G94, G961, G971: Feedrate in mm/min or inch/min or °/min
  - G95, G96, G97: Revoluional feedrate in mm/rev or inch/rev

---

Basic Functions

Function Manual, 01/2015, 6FC5397-0BP40-5BA2
17.4.10 Influencing the single axis dynamic response

Single axes

Single axes can be programmed in the part program, in synchronized actions and via the PLC:

- **Part program:**
  - POS[<axis>]=...
  - POSA[<axis>]=...
  - SPOS[<axis>]=...
  - SPOSA[<axis>]=...
  - OS[<axis>]=...
  - OSCILL[<axis>]=...

- **Synchronized actions:**
  - EVERY ... DO
  - POS[<axis>]=...
  - SPOS[<spindle>]=...
  - MOV[<axis>]=...

- **PLC:**
  - FC18

---

**Note**

The feedrate programmed under FB must be greater than zero.

If no traversing motion is programmed in the block (e.g. computation block), the FB has no effect.

If no explicit feed for chamfering/rounding is programmed, then the value of FB also applies for any contour element chamfering/rounding in this block.

Simultaneous programming of FB and FD (handwheel travel with feedrate override) or F (modal path feedrate) is not possible.
Dynamic response

The dynamic response of an axis is influenced by:

- **MD32060 $MA_POS_AX_VELO (positioning axis velocity)**
  The effective positioning axis velocity can be changed:
  - Part program / synchronized action: Axial feedrate FA or percentage feedrate override OVRA
  - PLC: Specification of FRate or overwriting the axial override

- **MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration)**
  The effective maximum axis acceleration can be changed:
  - Part program indirectly: Writing the machine data with subsequent NewConfig
  - Part program directly: Percentage acceleration override ACC
  - Synchronized actions indirectly: Writing the machine data and initiating an ASUB for the activation of NewConfig
  - Synchronized actions directly: Percentage acceleration override ACC (cannot be preset by the PLC).

  Via the PLC, the same options apply as in synchronized actions.

- **Part program commands: BRISKA, SOFTA, DRIVEA, JERKA**
  Cannot be programmed in synchronized actions (only indirectly via ASUB).
  Cannot be specified by the PLC (only indirectly via ASUB).

- **Active servo parameter set**
  The active parameter set can be changed:
  - Part program / synchronized action: SCPARA
  - PLC: DB31, … DBX9.0-2 (controller parameter set)

  For detailed information on the servo parameter sets, see "Parameter sets of the position controller (Page 361)".

Note

**Dynamic response changes**

Dynamic response changes made in the part program do not affect command or PLC axis motion. Dynamic response changes made in synchronized actions have no effect on traversing motion programmed in the part program.

**Feedforward control**

The type of feedforward control and the path axes that should be traversed with feedforward control can be directly programmed in the part program using FFWON/FFWOF. In synchronized actions and from the PLC, programming is only possible indirectly via an ASUB.

**Percentage acceleration override (ACC)**

In a part program or synchronized action, the acceleration specified in machine data: MD32300 $MA_MAX_AX_ACCEL (maximum axis acceleration) can be changed in a range from 0% – 200% using the ACC command.
Syntax:
ACC[<axis>]=<value>

Meaning:
ACC: Keyword for the programming of the percentage acceleration override
<aaxis>: Name of the channel axis or spindle
<value>: Acceleration change in percent relative to MD32300
Range of values: 0 ... 200

The actual axial acceleration value can be read via the system variable $AA_ACC. It is determined by:

$AA_ACC[<axis>] = (MD32300 $MA_MAX_AX_ACCEL[<axis>]) * ACC[<axis>] / 100

MD32320 $MA_DYN_LIMIT_RESET_MASK can be used to specify the initial setting of the value programmed with ACC for a channel reset or end of part program M30.

Note
The acceleration override programmed with ACC can be read using the system variable $AA_ACC. However, $AA_ACC is read in the part program at a different time than when reading in a synchronized action.

The system variables $AA_ACC only contain the value programmed in the part program with ACC if, in the meantime, the acceleration override was not changed by programming ACC in a synchronized action. The same applies for the reverse situation.

Percentage acceleration override and main run axes
Depending on whether the system variable $AA_ACC is read in the part program or synchronized action, the value for the acceleration override programmed with ACC is output for the NC axes or main run axes (command axes, PLC axes, asynchronous oscillating axes, etc.).

For correct results, system variable $AA_ACC must therefore always be read at the same location (part program or synchronized action) from where the acceleration override was programmed with ACC.

Examples:

**Writing ACC in a part program:**

N80 G01 POS[X]=100 FA[X]=1000 ACC[X]=90 IPOENDA[X]

**Writing ACC in a synchronized action:**


**Writing ACC and reading $AA_ACC in a part program:**

```
ACC[X]=50 ; writing
RO=$AA_ACC[X] ; reading
```
Writing ACC and reading $AA_ACC in a part program:

```plaintext
IF (RO <> $MA_MAX_AX_ACCEL[X] * 0.5) ; checking
  SETAL(61000)
ENDIF
```

Writing ACC and reading $AA_ACC in a synchronized action:

```plaintext
WHEN TRUE DO ACC[X]=25 R0=$AA_ACC[X]; writing and reading
  G4 F1

IF (RO <> $MA_MAX_AX_ACCEL[X] * 0.25) ; checking
  SETAL(61001)
ENDIF
```

end-of-motion criterion for single axes

Similar to the block change criterion for path interpolation (G601, G602, G603) the end-of-motion criterion for traversing motion of individual axes can be programmed in part programs / synchronized actions:

<table>
<thead>
<tr>
<th>Program command</th>
<th>End-of-motion criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINEA[&lt;axis&gt;]</td>
<td>&quot;Exact stop fine&quot;</td>
</tr>
<tr>
<td>COARSEA[&lt;axis&gt;]</td>
<td>&quot;Exact stop coarse&quot;</td>
</tr>
<tr>
<td>IPOENDA[&lt;axis&gt;]</td>
<td>&quot;Interpolator stop&quot; (IPO stop)</td>
</tr>
</tbody>
</table>

The most recently programmed value is kept after the end of program or NC reset.

The effective end-of-motion criterion can be read using the axis-specific system variable $AA_MOTEND.

Note

Depending on whether the system variable $AA_MOTEND is read in the part program or synchronized action, it contains the value for the NC axes or the main run axes.

Example:

**Part program:**

```plaintext
N80 G01 POS[X]=100 FA[X]=1000 ACC[X]=90 COARSEA[X]
```

**Synchronized action:**

```plaintext
```

References:

For further information on block changes and end-of-motion criteria for FINEA, COARSEA and IPOENDA, see:

Function Manual, Extended Functions; Positioning Axes (P2), Section: Block change
Programmable servo parameter set (SCPARA)

In the part program / synchronized action, the servo parameter set can be specified using SCPARA.

Syntax

SCPARA[<axis>] = <parameter set number>

Meaning

SCPARA: Keyword for the activation of the specified servo parameter set
<axis>: Name of the channel axis
<parameter set number>: Number of the servo parameter set to be activated

Note

The activation of the parameter set specified using SCPARA can be suppressed from the PLC user program:

DB31,… DBX9.3 = 1 (parameter set specification disabled by SCPARA)

In this case, no message is displayed.

The number of the active parameter set can be read using the system variable $AA_SCPAR.

Supplementary conditions

Different end-of-motion criteria

Different end-of-motion criteria will affect how quickly or slowly part program blocks are completed. This can have side effects for technology cycles and PLC user parts.

Parameter set change

The PLC user program must be expanded if the servo parameter set is to be changed both inside a part program or synchronized action and the PLC.

Power On

After POWER ON, the following initial settings are made:

- Percentage acceleration override for all single-axis interpolations: 100%
- End-of-motion criterion for all single-axis interpolations: FINEA
- Servo parameter set: 1

Mode change

When the operating mode is changed from AUTOMATIC to JOG, the programmed dynamic response changes remain valid.
17.5 Supplementary conditions

Reset
With reset, the last programmed value remains for the part program specifications. The settings for main-run interpolations do not change.

Block search
The last end-of-motion criterion programmed for an axis is collected and output in an action block. The last block with a programmed end-of-motion criterion that was processed in the search run serves as a container for all programmed end-of-motion criteria for all axes.

Unit of measurement
The valid unit of measurement of the feedrates depends on the set measuring system and the entered axis type:

MD10240 $MN_SCALING_SYSTEM_IS_METRIC (basic system of the control metric/inch)
MD30300 $MA_IS_ROT_AX (rotary or linear axis)

Initial setting for the feedrate type
The initial setting for the feedrate type is specified in the machine data:

MD20150 $MC_GCODE_RESET_VALUES (initial setting of the G groups)
The default setting is G94.
The initial setting of the feedrate type is only displayed when a part program is started.

Effectiveness after reset
Whether the last programmed F, FA, OVR, OVRA values are also active after reset depends on the setting in the machine data:

MD22410 $MC_F_VALUES_ACTIVE_AFTER_RESET (F function is active even after reset)

Spindle positioning
With active G95, G96, G961, G97, G971, G33, G34, G35 spindle positioning should not be performed, because the derived path feedrate following spindle positioning = 0. If the programmed axis position has not then been reached, the block cannot be completed.
17.6 Data lists

17.6.1 Machine data

17.6.1.1 NC-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10651</td>
<td>IPO_PARAM_THREAD_NAME_TAB</td>
<td>Name of the interpolation parameters for convex threads</td>
</tr>
<tr>
<td>10704</td>
<td>DRYRUN_MASK</td>
<td>Activation of dry run feedrate</td>
</tr>
<tr>
<td>10710</td>
<td>PROG_SD_RESET_SAVE_TAB</td>
<td>Setting data to be updated</td>
</tr>
<tr>
<td>11410</td>
<td>SUPPRESS_ALARM_MASK</td>
<td>Mask for suppressing special alarms</td>
</tr>
<tr>
<td>11550</td>
<td>STOP_MODE_MASK</td>
<td>Defines the stop behavior</td>
</tr>
<tr>
<td>12000</td>
<td>OVR_AX_IS_GRAY_CODE</td>
<td>Axis feedrate override switch, graycoded</td>
</tr>
<tr>
<td>12010</td>
<td>OVR_FACTOR_AX_SPEED</td>
<td>Evaluation of the axis feed override switch</td>
</tr>
<tr>
<td>12020</td>
<td>OVR_FEED_IS_GRAY_CODE</td>
<td>Path feed override switch, graycoded</td>
</tr>
<tr>
<td>12030</td>
<td>OVR_FACTOR_FEEDRATE</td>
<td>Evaluation of the path feed override switch</td>
</tr>
<tr>
<td>12040</td>
<td>OVR_RAPID_IS_GRAY_CODE</td>
<td>Rapid traverse override switch, graycoded</td>
</tr>
<tr>
<td>12050</td>
<td>OVR_FACTOR_RAPID_TRA</td>
<td>Evaluation of the rapid traverse override switch</td>
</tr>
<tr>
<td>12060</td>
<td>OVR_SPIND_IS_GRAY_CODE</td>
<td>Spindle override switch, graycoded</td>
</tr>
<tr>
<td>12070</td>
<td>OVR_FACTOR_SPIND_SPEED</td>
<td>Evaluation of the spindle override switch</td>
</tr>
<tr>
<td>12080</td>
<td>OVRREFERENCE_IS_PROG_FEED</td>
<td>Override reference velocity</td>
</tr>
<tr>
<td>12082</td>
<td>OVRREFERENCE_IS_MIN_FEED</td>
<td>Defining the reference of the path override</td>
</tr>
<tr>
<td>12090</td>
<td>OVR_FUNCTION_MASK</td>
<td>Selection of override specifications</td>
</tr>
<tr>
<td>12100</td>
<td>OVR_FACTOR_LIMIT_BIN</td>
<td>Limit for binarycoded override switch</td>
</tr>
<tr>
<td>12200</td>
<td>RUN_OVERRIDE_0</td>
<td>Traversing with override 0</td>
</tr>
<tr>
<td>12202</td>
<td>PERMANENT_FEED</td>
<td>Fixed feedrates for linear axes</td>
</tr>
<tr>
<td>12204</td>
<td>PERMANENT_ROT_AX_FEED</td>
<td>Fixed feedrates for rotary axes</td>
</tr>
</tbody>
</table>

17.6.1.2 Channel-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20100</td>
<td>DIAMETER_AX_DEF</td>
<td>Geometry axes with transverse axis functions</td>
</tr>
<tr>
<td>20150</td>
<td>GCODE_RESET_VALUES</td>
<td>Initial setting of the G groups</td>
</tr>
<tr>
<td>20172</td>
<td>COMPRESS_VELO_TOL</td>
<td>Maximum permissible deviation from path feed for compression</td>
</tr>
<tr>
<td>20200</td>
<td>CHFRND_MAXNUM_DUMMY_BLOCKS</td>
<td>Empty blocks with phase/radii</td>
</tr>
<tr>
<td>20201</td>
<td>CHFRND_MODE_MASK</td>
<td>Behavior for chamfer/rounding</td>
</tr>
<tr>
<td>20750</td>
<td>ALLOW_GO_IN_G96</td>
<td>G0 logic for G96, G961</td>
</tr>
<tr>
<td>21200</td>
<td>LIFTFAST_DIST</td>
<td>Traversing path for fast retraction from the contour</td>
</tr>
</tbody>
</table>
### 17.6 Data lists

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC___</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>21220</td>
<td>MULTFEED_ASSIGN_FASTIN</td>
<td>Assignment of input bytes of NCK I/O for &quot;Multiple feedrate values in one block&quot;</td>
</tr>
<tr>
<td>21230</td>
<td>MULTFEED_STORE_MASK</td>
<td>Storage behavior for the &quot;Multiple feedrate values in one block&quot; function</td>
</tr>
<tr>
<td>22240</td>
<td>AUXFU_F_SYNC_TYPE</td>
<td>Output timing of F functions</td>
</tr>
<tr>
<td>22410</td>
<td>F_VALUES_ACTIVE_AFTER_RESET</td>
<td>F function active after reset</td>
</tr>
</tbody>
</table>

#### 17.6.1.3 Axis/Spindle-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA___</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>30300</td>
<td>IS_ROT_AX</td>
<td>Rotary axis/spindle</td>
</tr>
<tr>
<td>32000</td>
<td>MAX_AX_VELO</td>
<td>Maximum axis velocity</td>
</tr>
<tr>
<td>32060</td>
<td>POS_AX_VELO</td>
<td>Initial setting for positioning axis velocity</td>
</tr>
<tr>
<td>32300</td>
<td>MAX_AX_ACCEL</td>
<td>Axis acceleration</td>
</tr>
<tr>
<td>32320</td>
<td>DYN_LIMIT_RESET_MASK</td>
<td>Reset behavior of dynamic limits</td>
</tr>
<tr>
<td>34990</td>
<td>ENC_ACTIVAL_SMOOTH_TIME</td>
<td>Smoothing time constant for actual values</td>
</tr>
<tr>
<td>35100</td>
<td>SPIND VELO LIMIT</td>
<td>Maximum spindle speed</td>
</tr>
<tr>
<td>35130</td>
<td>GEAR_STEP_MAX_VELO_LIMIT</td>
<td>Maximum speed of gear stage</td>
</tr>
<tr>
<td>35140</td>
<td>GEAR_STEP_MIN_VELO_LIMIT</td>
<td>Minimum speed of gear stage</td>
</tr>
<tr>
<td>35160</td>
<td>SPIND_EXTERN_VELO_LIMIT</td>
<td>Spindle-speed limitation via PLC</td>
</tr>
</tbody>
</table>

#### 17.6.2 Setting data

#### 17.6.2.1 Channel-specific setting data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $SC___</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>42000</td>
<td>THREAD_START_ANGLE</td>
<td>Start angle for thread</td>
</tr>
<tr>
<td>42010</td>
<td>THREAD_RAMP_DISP</td>
<td>Acceleration behavior of axis when thread cutting</td>
</tr>
<tr>
<td>42100</td>
<td>DRY_RUN_FEED</td>
<td>Dry run feedrate</td>
</tr>
<tr>
<td>42101</td>
<td>DRY_RUN_FEED_MODE</td>
<td>Dry run feed mode</td>
</tr>
<tr>
<td>42110</td>
<td>DEFAULT_FEED</td>
<td>Default value for path feed</td>
</tr>
<tr>
<td>42600</td>
<td>JOG_FEED_PER_REV_SOURCE</td>
<td>Revolutional feedrate control in the JOG mode</td>
</tr>
<tr>
<td>43300</td>
<td>ASSIGN_FEED_PER_RES_SOURCE</td>
<td>Revolutionl feedrate for positioning axes/spindles</td>
</tr>
</tbody>
</table>
17.6.2.2 Axis/spindle-specific setting data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $SA_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>43210</td>
<td>SPIND_MIN_VELO_G25</td>
<td>Programmed spindle speed limiting G25</td>
</tr>
<tr>
<td>43220</td>
<td>SPIND_MAX_VELO_G26</td>
<td>Programmed spindle speed limiting G26</td>
</tr>
<tr>
<td>43230</td>
<td>SPIND_MAX_VELO_LIMS</td>
<td>Spindle speed limiting with G96</td>
</tr>
</tbody>
</table>

17.6.3 Signals

17.6.3.1 Signals to channel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate dry run feedrate</td>
<td>DB21, ... .DBX0.6</td>
<td>DB320x.DBX0.6</td>
</tr>
<tr>
<td>Feedrate override</td>
<td>DB21, ... .DBB4</td>
<td>DB320x.DBB4</td>
</tr>
<tr>
<td>Rapid traverse override</td>
<td>DB21, ... .DBB5</td>
<td>DB320x.DBB5</td>
</tr>
<tr>
<td>Feed disable</td>
<td>DB21, ... .DBX6.0</td>
<td>DB320x.DBX6.0</td>
</tr>
<tr>
<td>Rapid traverse override active</td>
<td>DB21, ... .DBX6.6</td>
<td>DB320x.DBX6.6</td>
</tr>
<tr>
<td>Feedrate override active</td>
<td>DB21, ... .DBX6.7</td>
<td>DB320x.DBX6.7</td>
</tr>
<tr>
<td>Feed stop, geometry axis 1</td>
<td>DB21, ... .DBX12.3</td>
<td>DB320x.DBX1000.3</td>
</tr>
<tr>
<td>Feed stop, geometry axis 2</td>
<td>DB21, ... .DBX16.3</td>
<td>DB320x.DBX1004.3</td>
</tr>
<tr>
<td>Feed stop, geometry axis 3</td>
<td>DB21, ... .DBX20.3</td>
<td>DB320x.DBX1008.3</td>
</tr>
</tbody>
</table>

17.6.3.2 Signals from channel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dry run feedrate selected</td>
<td>DB21, ... .DBX24.6</td>
<td>DB170x.DBX0.6</td>
</tr>
<tr>
<td>Feedrate override for rapid traverse selected</td>
<td>DB21, ... .DBX25.3</td>
<td>DB170x.DBX1.3</td>
</tr>
<tr>
<td>Activate fixed feedrate 1 for path/geometry axes</td>
<td>DB21, ... .DBX29.0</td>
<td>DB320x.DBX13.0</td>
</tr>
<tr>
<td>Activate fixed feedrate 2 for path/geometry axes</td>
<td>DB21, ... .DBX29.1</td>
<td>DB320x.DBX13.1</td>
</tr>
<tr>
<td>Activate fixed feedrate 3 for path/geometry axes</td>
<td>DB21, ... .DBX29.2</td>
<td>DB320x.DBX13.2</td>
</tr>
<tr>
<td>Activate fixed feedrate 4 for path/geometry axes</td>
<td>DB21, ... .DBX29.3</td>
<td>DB320x.DBX13.3</td>
</tr>
<tr>
<td>Dry run feedrate active</td>
<td>DB21, ... .DBX318.6</td>
<td>DB330x.DBX4002.6</td>
</tr>
</tbody>
</table>

17.6.3.3 Signals to axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feed/spindle override</td>
<td>DB31, ... .DBB0</td>
<td>DB380x.DBB0</td>
</tr>
<tr>
<td>Override active</td>
<td>DB31, ... .DBX1.7</td>
<td>DB380x.DBX1.7</td>
</tr>
</tbody>
</table>
## 17.6 Data lists

### 17.6.3.4 Signals from axis/spindle

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revolutinal feedrate active</td>
<td>DB31, ... DBX62.2</td>
<td>DB390x.DBX2.2</td>
</tr>
<tr>
<td>F function for positioning axis</td>
<td>DB31, ... DBB81</td>
<td>-</td>
</tr>
<tr>
<td>Programmed speed too high</td>
<td>DB31, ... DBX83.1</td>
<td>DB390x.DBX2001.1</td>
</tr>
</tbody>
</table>
18.1 Brief description

Calculating tool compensation data
The SINUMERIK 840D sl controller can be used to calculate the following tool compensation data:

- Length compensation
- Radius compensation
- Storage of tool data in a flexible tool offset memory:
  - Tool identification with T numbers from 0 to 32000
  - Definition of a tool with a maximum of 9 cutting edges
  - Cutting edge described by up to 25 tool parameters
- Tool selection selectable: Immediate or via selectable M function
- Tool radius compensation:
  - Selection and deselection strategy configurable: Normal or contour-related
  - Compensation active for all interpolation types:
    Linear
    Circle
    Helical
    Spline
    Polynomial
    Involute
  - Compensation at outer corners selectable:
    Transition circle/ellipse (G450) or equidistant intersection (G451)
  - Parameter-driven adaptation of G450/G451 functions to the contour
  - Free traversing on outer corners with G450 and DISC parameter
  - Number of dummy blocks without axis motion selectable in the compensation plane
  - Collision detection selectable:
    Possible contour violations are detected predictively, if:
    - Path is shorter than tool radius
    - Width of an inside corner is shorter than the tool diameter
  - Keep tool radius compensation constant
  - Intersection procedure for polynomials
Toolholder with orientation capability

This function permits the machining of inclined surfaces with allowance for tool length compensation, provided that the kinematics of the toolholder (without NC axes) permits a static orientation of the tool. The more complex 5-axis transformation is not required for this case.

Reference:
Function Manual, Special Functions; Multi-Axis Transformations (F2)

Appropriate selection of the tool data and toolholder data describes the kinematics for the controller such that it can make allowance for the tool length compensation. The controller can take some of the description data direct from the current frame.

Note

Please refer to the following documentation for further information on tools and tool compensations and a full technical description of the general and specific programming features for tool compensation (TLC and TRC):

References:
Programming Manual, Fundamentals

Flat/unique D number structure

Compensations can be selected via unique D numbers with management function.

Special handling of tool compensations

The evaluation of signs can be controlled for tool length and wear by the setting data:
SD42900 $SC_MIRROR_TOOL_LENGTH (sign change tool length when mirroring)
SD42960 $SC_TOOL_TEMP_COMP (temperature compens. regarding tool).

The same applies to the response of the wear components when mirroring geometry axes or changing the machining plane via setting data.

References:
Programming Manual, Fundamentals, Tool Offsets
G461/G462

In order to enable the solid machining of inside corners in certain situations with the activation and deactivation of tool radius compensation, commands G461 und G462 have been introduced and the approach/retraction strategy has thus been extended for tool radius compensation.

- **G461**
  If no intersection is possible between the last TRC block and a previous block, the controller calculates an intersection by extending the offset curve of this block with a circle whose center point coincides with the end point of the noncorrected block, and whose radius is equal to the tool radius.

- **G462**
  If no intersection is possible between the last TRC block and a previous block, the controller calculates an intersection by inserting a straight line at the end point of the last block with tool radius compensation (the block is extended by its end tangent).

**Changing from G40 to G41/G42**

The change from G40 to G41/G42 and vice versa is no longer treated as a tool change for tools with relevant tool point direction (turning and grinding tools).

**Tool compensation environments**

Functions which enable the following actions in relation to the current states of tool data are available in SW 7.1:

- Save
- Delete
- Read
- Modify

Some of the functions were previously implemented in measuring cycles. They are now universally available.

A further function can be used to determine information about the assignment of the tool lengths of the active tool to the abscissa, ordinate and applicate.

### 18.2 Tool

#### 18.2.1 General

**Select tool**

A tool is selected in the program with the T function.
Whether the new tool will be loaded immediately by means of the T function depends on the setting in the machine data:

MD22550 $MC_TOOL_CHANGE_MODE (new tool compensation with M function)

Change tool immediately

MD22550 $MC_TOOL_CHANGE_MODE = 0 (new tool compensation with M function).
The new tool is changed immediately with the T function.
This setting is used mainly for turning machines with tool revolver.

Change tool with M06

MD22550 $MC_TOOL_CHANGE_MODE = 1 (new tool compensation with M function).
The new tool is prepared for changing with the T function.
This setting is used mainly on milling machines with a tool magazine, in order to bring the new tool into the tool change position without interrupting the machining process.
The old tool is removed from the spindle and the new tool is loaded into the spindle with the entered M function in the machine data:
- MD22560 $MC_TOOL_CHANGE_M_CODE (M function for tool change)
This tool change must be programmed with the M function M06, in accordance with DIN 66025.
The next tool is preselected with the machine data:
- MD20121 $MC_TOOL_PRESEL_RESET_VALUE (Preselected tool at RESET)
Its tool length compensation values must be considered at RESET and powerup according to machine data:
- MD20110 $MC_RESET_MODE_MASK (Determination of control default settings after RESET/TP end).

Value range of the T function or the tool number

The T function or the tool number can assume the following integer numbers:
- Minimum value: T0 (no tool)
- Maximum value: T32000 (tool with number 32000)
Tool cutting edge D

A tool Tx can be assigned various tool cutting edges D1 ... Dn.

Figure 18-1 Example of a tool Tx with 9 cutting edges (D1 to D9)

Maximum number of tool cutting edges per tool

The maximum number of tool cutting edges (D numbers) that can be defined for each tool, is specified using the following machine data:

- MD18106 $MN_MM_MAX_CUTTING_EDGE_PERTOOL

D function

The selection of the tool cutting edge 1 ... n required for machining is realized by programming the appropriate D function, e.g. in the part program with D1 ... Dn. The selected tool cutting edge always refers to the currently active tool. A tool cutting edge without active tool (actual T number is T0 ⇒ no tool selected), is not effective.

All tool offsets of the active tool Tx are deselected using D0.

Selecting the cutting edge when changing tool

When selecting a new tool using a new T number, and loading this tool, the following options are available to select the cutting edge:

- The cutting edge number Dx is programmed.
- The cutting edge number is defined by the machine data:
  MD20270 $MC_CUTTING_EDGE_DEFAULT =<value> (basic position of the tool cutting edge without programming)

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No automatic cutting edge selection in accordance with M06</td>
</tr>
<tr>
<td>≠ 0</td>
<td>Number of the cutting edge, which is selected in accordance with M06</td>
</tr>
<tr>
<td>-1</td>
<td>The cutting edge number of the old tool is retained and is also selected for the new tool, in accordance with M06</td>
</tr>
</tbody>
</table>
Activating the tool offset

D1 to Dn activate the tool compensation for a cutting edge on the active tool. Tool length compensation and tool radius compensation can be activated at different times:

- **Tool length compensation (TLC)** is performed on the first traversing motion of the axis, on which the TLC is to act.
  This traversing motion must be a linear interpolation (G0, G1, POS, POSA) or polynomial interpolation (POLY). If the POS/POSA axis is one of the active geometry axes, the tool length compensation is applied with the first axis motion in which the WLK is supposed to act.

- **Tool radius compensation (TRC)** becomes active when G41/G42 is programmed in the active plane (G17, G18, or G19).
  In the NC block in which the tool radius correction is activated with G41/G42, G0 (rapid traverse) or G1 (linear interpolation) must be active and at least one geometry axis of the active working plane must be programmed.
  If only one geometry axis is specified when switching on, the last position of the second geometry axis is automatically supplemented and then traversed in both axes.
  If the axes are not geometry axes, then they must be transformed into geometry axes using GEOAX programming.
  The selection of tool radius compensation with G41/G42 is only permitted in a program block with G0 (rapid traverse) or G1 (linear interpolation).

### 18.2.2 Compensation memory structure

**Tool compensation memory size**

Each channel can have a dedicated tool compensation memory (TO unit).

Which tool compensation memory exists for the relevant channel is set with the machine data:

MD28085 $MC_MM_LINK_TOA_UNIT (Assignment of TO unit to a channel).

The maximum number of tool cutting edges for all tools managed by the NCK is set with the machine data:

MD18100 $MN_MM_NUM_CUTTING_EDGES_IN_TOA (number of tool cutting edges in NCK).

**Tools**

The TO memory consists of tools numbered T1 to T32000.

Each tool can be set up via TOA files or individually, using the "New tool" soft key.

Compensation values not required must be assigned the value zero. (this is the default setting
when the offset memory is created): The individual values in the offset memory (tool parameters) can be read and written from the program using system variables.

Note

The tools (T1 to T32000) do not have to be stored in ascending order or contiguously in the tool compensation memory, and the first tool does not have to be assigned number T1.

Tool cutting edges

Each tool can have up to 9 cutting edges (D1 to D9). The first cutting edge (D1) is set up automatically when a new tool is loaded in the tool compensation memory. Other cutting edges (up to 8) are set up consecutively and contiguously using the “New cutting edge” soft key. A different number of tool cutting edges can assigned to each tool in this way.

Figure 18-2  Example of a tool compensation memory structure for 2 channels
18.2.3 Calculating the tool compensation

D No.

The D no. is sufficient for calculating the tool compensations (can be set via MD).

![D No. Table]

Programming

The above compensation block is to be calculated in the NC.

Part program call:

```
... 
Dn
```

18.2.4 Address extension for NC addresses T and M

MD20096

Whether also with tool management not activated, the address extension of T and M is to be interpreted as spindle number, can be set through the machine data:

MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO (spindle number as address extension).

The same rules then apply to the reference between the D number and T number as when the "Tool management" function is active.

Effect on the D number

A compensation data set is determined by the D number.

The D address cannot be programmed with an address extension.

The evaluation of the D address always refers to the currently active tool.
The programmed D address refers to the active tool in relation to the master spindle (same as for tool management function), when machine data is set:

MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = TRUE (spindle number as address extension).

**Effect on the T number**

If the "Tool management" function is active, the values programmed with reference to the master spindle (or master toolholder) are displayed as programmed/active T numbers.

If tool management is not active, all programmed T values are displayed as programmed/active, regardless of the programmed address extension.

Only the T value programmed in relation to the master spindle is shown as programmed/active, when:

MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = TRUE (spindle number as address extension).

**Example**

The example below shows the effect of MD20096.

Two spindles are considered. Spindle 1 is the master spindle. M6 was defined as the tool change signal.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>5</td>
</tr>
<tr>
<td>M1</td>
<td>6</td>
</tr>
<tr>
<td>T2</td>
<td>50</td>
</tr>
<tr>
<td>M2</td>
<td>6</td>
</tr>
<tr>
<td>D4</td>
<td></td>
</tr>
</tbody>
</table>
If tool management is active, \( D4 \) refers to tool "5". T2=50 defines the tool for the secondary spindle, whose tool does not influence the path compensation. The path is determined exclusively by the tool programmed for the master spindle.

- \( D4 \) relates to tool "50" without active tool management and with the machine data: MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = \text{FALSE} \) (significance of address extension on T, M tool change). The address extensions of neither T nor M are evaluated in the NCK. Each tool change command defines a new path compensation.

- \( D4 \) relates to tool "5" (as when tool management is active) without active tool management and with the machine data: MD20096 $MC_T_M_ADDRESS_EXT_IS_SPINO = \text{TRUE} \). Address extension 1 (T1= ..., M1= ...) addresses the master spindle.

**Note**

Previously, when tool management was not activated, each tool change command (programmed with T or M) caused the tool compensation to be recalculated in the path. The address extension is not defined further by this operation. The significance of the extension is defined by the user (in the PLC user program).

### 18.2.5 Free assignment of D numbers

"Relative" D numbers

In the NC it is possible to manage the D numbers as "relative" D numbers for the tool compensation data sets. The corresponding D numbers are assigned to each T number.
Functions

Expansions to functions when assigning D numbers:

- The maximum permitted D numbers are defined via the machine data:
  MD18105 $MN_MM_MAX_CUTTING_EDGE_NO (max. value of the D numbers (DRAM))
  The default value is 9, in order to maintain compatibility with existing applications.
- The number of cuts (or the offset data sets) for each tool can be defined via the machine data:
  MD18106 $MN_MM_MAX_CUTTING_EDGE_PERTOOL (max. number of the D numbers per tool (DRAM))
  This allows you to customize the number of cutting edges to be configured for each tool to
  the actual number of real cutting edges for monitoring purposes.
- It is also possible to rename D numbers in the NCK and thus to allocate any D numbers to
  the cutting edges.

Note

In addition to relative D number allocation, the D numbers can also be assigned as "flat"
or "absolute" D numbers (1-32000) without a reference to a T number (within the "Flat D
number structure" function).

Cutting edge number CE

When you rename D numbers, the information in the tool Catalog detailing the numbers defined
for these cutting edges is lost. It is, therefore, impossible to determine, following renaming,
which cutting edge of the Catalog is being referenced.

Since this information is required for retooling procedures, a cutting edge number CE has been
introduced for each cutting edge. This number remains stored when the D number is renamed.

The D number identifies the cutting edge compensation in the part program. This compensation
number D is administered separately from the cutting edge number CE (the number in the tool
Catalog). Any number can be used. The number is used to identify a compensation in the part
program and on the display.

The CE number identifies the actual physical cutting edge during retooling. The cutting edge
number CE is not evaluated by the NCK on compensation selection during a tool change (only
available via the OPI).

The cutting edge number CE is defined with system variable $TC_DPCE[t,d]:

- t stands for the internal T number.
- d stands for the D number.

Write accesses are monitored for collisions, i.e. all cutting edge numbers of a tool must be
different. The variable $TC_DPCE is a component of the cutting edge parameter data set
$TC_DP1 to $TC_DP25.

It is only practical to parameterize $TC_DPCE if the maximum cutting edge number (MD18105)
is greater than the maximum number of cutting edges per tool (MD18106).

In this case, the default cutting edge number is the same as the classification number of the
cutting edge. Compensations of a tool are created starting at number 1 and are incremented
up to the maximum number of cutting edges per tool (MD18106).
The cutting edge number CE is the same as the D number (in compatibility with the behavior till now) if:

MD18105 ≤ MD18106.

A read operation returns CE=D. A write operation is ignored without an alarm message.

**Note**

The compensation values $TC_DP1$ to $TC_DP25$ of the active tool compensation can be read with system variable $SP_AD[n]$, where n=1 to 25. The CE cutting edge number of the active compensation is returned with n=26.

### Commands

If the maximum cutting edge number is **higher** than the maximum number of cutting edges for each tool, then the commands listed in the following table, are available

- (MD18105 $MN_MM_MAX_CUTTING_EDGE_NO) > (MD18106 $MN_MM_MAX_CUTTING_EDGE_PERTOOL)

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHKDNO</td>
<td>Checks the uniqueness of the available D numbers.</td>
</tr>
<tr>
<td>GETDNO</td>
<td>Determines the D number for the cutting edge of a tool.</td>
</tr>
<tr>
<td>SETDNO</td>
<td>Sets or changes the D number of the CE cutting edge of tool T.</td>
</tr>
<tr>
<td>GETACTTD</td>
<td>Determines the associated T number for an absolute D number.</td>
</tr>
<tr>
<td>DZERO</td>
<td>Marks all D numbers of the TO unit as invalid.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHKDNO</td>
<td>Checks the uniqueness of the available D numbers.</td>
</tr>
<tr>
<td>GETDNO</td>
<td>Determines the D number for the cutting edge of a tool.</td>
</tr>
<tr>
<td>SETDNO</td>
<td>Sets or changes the D number of the CE cutting edge of tool T.</td>
</tr>
<tr>
<td>GETACTTD</td>
<td>Determines the associated T number for an absolute D number.</td>
</tr>
<tr>
<td>DZERO</td>
<td>Marks all D numbers of the TO unit as invalid.</td>
</tr>
</tbody>
</table>

This command is used for support during retooling.

Compensation data sets tagged with this command are no longer verified by the CHKDNO language command. These data sets can be accessed again by setting the D number once more with SETDNO.
Note

If the maximum cutting edge number is less than the maximum number of cutting edges for each tool, then the commands listed in the table are ineffective. This default setting is in the system for reasons of compatibility.

The individual commands are described in detail in:

References:
Programming Manual Fundamentals

Activation

In order to work with unique D numbers and, therefore, with the defined language commands, it must be possible to name D numbers freely for the tools.

The following conditions must be fulfilled for this purpose:

- MD18105 > MD18106
- The 'flat D number' function is not activated.

MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE (type of D number programming (SRAM)).

Examples

MD18105 $MN_MM_MAX_CUTTING_EDGE_NO = 1 (max. value of the D numbers)
A maximum of one compensation can be defined per tool (with D number = 1).

Note

When the "Flat D numbers" function is active, only one D compensation can be defined in the TO unit.

MD18105 $MN_MM_MAX_CUTTING_EDGE_NO = 9999
Tools can be assigned unique D numbers.
Example: Assigning D (cutting edges) to T numbers (tools) and listing options for checking uniqueness:

- **Assignment:**
  - T No. 1 ⇒ assignment of D numbers 1, 2, 3
  - T No. 2 ⇒ assignment of D numbers 10, 20, 30, 40, 50
  - T No. 3 ⇒ assignment of D numbers 100, 200, 30 (error: 30 was entered instead of 300)

- **Check for uniqueness:**
  The D numbers of all tools defined within a TO unit must be unique. The check is made using the CHKDNO command. When no parameters are specified, all D numbers of all tools are checked with respect to one another. In this particular case, the return value == FALSE, as D30 is available in tool T2 and T3.

<table>
<thead>
<tr>
<th>Command</th>
<th>Test</th>
<th>Test result</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHKDNO</td>
<td>All D numbers of all tools are checked against all other tools</td>
<td>Return value == FALSE, as D=30 is duplicated.</td>
</tr>
<tr>
<td>CHKDNO (2, 3, 30)</td>
<td>D number 30 in tools 2 and 3</td>
<td>Return value == FALSE, as D=30 is duplicated.</td>
</tr>
<tr>
<td>CHKDNO (1, 3)</td>
<td>All D numbers of tools 1 and 3</td>
<td>Return value == TRUE although there is a collision between the D=30 of the third tool and D=30 of the second tool.</td>
</tr>
</tbody>
</table>

**Note**

**Cutting edges per tool**

If tools with n cutting edges are used, then the value of the machine data should be set to n. This prevents inadmissible cutting edges to be defined for a tool.

MD18106 $MN_MM_MAX_CUTTING_EDGE_PER_TOOL = n

**Programming examples**

**Renaming a D number**

The D number of cutting edge CE = 3 is to be renamed from 2 to 17. The following specifications apply:

- $TC_DPx[<tool Tn>, <cutting edge Dm>]
- Internal T number Tn = 1
- D number Dm = 2
- Tool with one cutting edge with:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_DP2[1, 2] = 120</td>
<td>Tool length = 120</td>
</tr>
<tr>
<td>$TC_DP3[1, 2] = 5.5</td>
<td>Tool radius = 5.5</td>
</tr>
<tr>
<td>$TC_DPCE[1, 2] = 3</td>
<td>Cutting edge number CE = 3</td>
</tr>
</tbody>
</table>

- MD18105 $MN_MM_MAX_CUTTING_EDGE_NO = 20 (max. value of the D numbers)

Within the part program, this compensation is programmed as standard with T1, ...D2.
You assign the current D number of cutting edge 3 to a variable (DNoOld) and define the variable DNoNew for the new D number:

```
Program code

def int DNoOld, DNoNew = 17
DNoOld = GETDNO( 1, 3 )
SETDNO( 1, 3, DNoNew )
```

The new D value 17 is then assigned to cutting edge CE=3.

Now the data for the cutting edge are addressed via D number 17, both via the system variable and in programming with the NC address D.

This compensation is now programmed in the part program with `T1, ....D17` and the data are addressed as follows:

```
Program code

$TC_DP2[ 1, 17 ] = 120
$TC_DP3[ 1, 17 ] = 5.5
$TC_DPCE[ 1, 17 ] = 3

; Cutting edge number CE
```

**Note**

If a further cutting edge has been defined for the tool, e.g. `$TC_DPCE[ 1, 2 ] = 1 ; = CE`, the D-number 2 of the cutting edge 1 cannot have the same name as the D-number of the cutting edge 3 i.e.: `SETDNO( 1, 1, 17)` returns the status = FALSE as return value.

**DZERO - Invalidate D numbers**

The activation of this command invalidates all D numbers of the tools in the TO unit. It is no longer possible to activate a compensation until valid D numbers are again available in the NCK. The D numbers must be reassigned using the SETDNO command.

The following tools must be defined (all with cutting edge number 1):

<table>
<thead>
<tr>
<th>Tool</th>
<th>D Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>D1</td>
</tr>
<tr>
<td>T2</td>
<td>D10</td>
</tr>
<tr>
<td>T3</td>
<td>D100</td>
</tr>
</tbody>
</table>

The following command is then programmed:

```
Program code

DZERO
```

If one of the compensations is now activated (e.g. with `T3 D100`), an alarm is generated, because `D100` is not currently defined.

The D numbers are redefined with:

```
Program code

SETDNO( 1, 1, 100 ) ; T=1, cutting edge 1 is assigned the (new) D number 100
SETDNO( 2, 1, 10 )  ; T=2, cutting edge 1 is assigned the (old) D number 10
SETDNO( 3, 1, 1 )   ; T=3, cutting edge 1 is assigned the (new) D number 1
```
Note
In the event of a power failure, the DZERO command can leave the NCK in an undefined state with reference to the D numbers. If this happens, repeat the DZERO command when the power is restored.

Operating principle of a retooling program
Let us assume you want to ensure that the required tools and cutting edges are available. The status of the tool-holding magazine of the NCK is arbitrary. The D numbers in the part programs for the new machining operation generally do not match the D numbers of the actual cutting edges. The retooling program can have the following appearance:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DZERO</td>
<td>; All D numbers of the TO unit are tagged as invalid.</td>
</tr>
<tr>
<td>....</td>
<td>; One or more loops over the locations of the magazine to check the tools and their cutting edge numbers.</td>
</tr>
<tr>
<td></td>
<td>; If a tool is found, which is still enabled ($TC_TP8) and has the required cutting edge number CE (GETDNO), the new D number is allocated to the cutting edge (SETDNO).</td>
</tr>
<tr>
<td>....</td>
<td>; Loading and unloading operations are performed.</td>
</tr>
<tr>
<td></td>
<td>; It is possible to work with the tool status 'to be unloaded' and 'to be loaded'.</td>
</tr>
<tr>
<td>CHKDNO</td>
<td>; Loading/unloading and the operation for renaming D numbers are complete.</td>
</tr>
<tr>
<td></td>
<td>; Individual tools and/or D numbers can be checked, and collisions can be handled automatically according to the return value.</td>
</tr>
</tbody>
</table>

18.2.6 Compensation block in case of error during tool change

MD22550
If a tool preparation has been programmed in the part program and the NCK detects an error (e.g., the data set for the programmed T number does not exist in the NCK), the user can assess the error situation and perform appropriate tasks, in order to subsequently resume machining.

The tool change may be programmed independently, depending on the machine data:

MD22550 $MC_TOOL_CHANGE_MODE (new tool compensation with M function).

MD22550 $MC_TOOL_CHANGE_MODE = 0

T= "T no." ; Tool preparation + tool change in one NC block, i.e., when T is programmed a new D compensation becomes active in the NCK (see machine data MD20270 $MC_CUTTING_EDGE_DEFAULT)
MD22550 $MC_TOOL_CHANGE_MODE = 1

| T= "T no." ; Tool preparation |
| M06 ; Change tool |
| ; (the number of the tool-change M code can be changed), |
| ; i.e., when M06 is programmed a new D compensation becomes |
| ; active in the NCK (see |
| ; machine data MD20270 $MC_CUTTING_EDGE_DEFAULT) |

The following problems can occur if tool management is not active:

- **D compensation data set missing**
- **Error in part program**

**Note**

The "tool not in magazine" problem cannot be detected since the NCK did not have access to any magazine information during tool compensation.

**D compensation data set missing**

Program execution is interrupted at the block containing the invalid D value (regardless of the value of machine data MD22550). The operator must either correct the program or reload the missing data set.

To do this, he needs the D number for the flat D number function, or otherwise the T number as well. These parameters are transferred when the alarm is triggered.

**Error in part program**

The options for intervention in the event of an error depend on how the tool change was programmed, defined via the machine data:

MD22550 $MC_TOOL_CHANGE_MODE (new tool compensation with M function).

**Tool change with T programming (MD22550 = 0)**

In this case, the "Compensation block" function available in the NCK is used. The NC program stops at the NC block in which a programmed T value error was detected. The "Compensation block" is executed again when the program is resumed.

The operator can intervene as follows:

- Correct the part program.
- Reload the missing cutting edge compensation data from the HMI.
- Include the missing cutting edge compensation data in the NCK using "Overstore".

Following operator intervention, the START key is pressed and the block, which caused the error, is executed again. If the error was corrected, the program is continued. Otherwise, an alarm is output again.

**Tool change with T and M06 programming (MD22550 = 1)**
In this case, an error is detected in the NC block containing the tool preparation (T programming), however this error is to be ignored initially. Processing continues until the tool change request (usually M06) is executed. The program is to stop at this point.

The programmed T address can contain any number of program lines ahead of the M06 command, or the two instructions can appear in different (sub)programs. For this reason, it is not usually possible to modify a block or a compensation block, which has already been executed.

The operator has the same options for intervention as with = 0.

Reloading of missing data is possible. In this case, however, T must be programmed with "Overstore".

If a program error has occurred, the line with the error cannot be corrected (Txx); only the line at which the program stopped and which generated the alarm can be edited. Only when machine data:

MD22562 $MC_TOOL_CHANGE_ERROR_MODE Bit0 = 1 (response on errors in tool change).

The sequence is as follows:

```
Txx ; Error! Data set with xx does not exist.
    ; Detect state; detect xx;
    ; continue in program

....

M06 ; Detect bit memory "xx missing" → output alarm,
    ; stop program
    ; Correct block with, e.g., Tyy M06, start,
    ; block Tyy M06 interpreted and OK.
    ; Machining continues.
```

The following occurs when this part of the program is executed again:

```
Txx ; Error! Data set with xx does not exist,
    ; Detect state; detect xx;
    ; continue in program

....

Tyy M06 ; Detect bit memory "xx missing" → cancel without further response,
    ; as Tyy M06 is correct → program does not stop (correct).
```

If necessary, the original point of the T call can be corrected after the end of the program. If the tool change logic on the machine cannot process this, the program must be aborted and the point of the error corrected.

If only one data set is missing, it is transferred to the NCK, Txx is programmed in "Overstore" and the program is subsequently resumed.
As in the case of "missing D number", the required parameter (T number) can be accessed by the user for "missing T number" via the appropriate alarm (17191).

**Note**
In order to enable program correction, it stops immediately at the faulty Txx block. The program text operation is also stopped when machine data:

MD22562 $MC_TOOL_CHANGE_ERROR_MODE Bit0=1 (response on errors in tool change).

### 18.2.7 Definition of the effect of the tool parameters

**MD20360**

The effect of the tool parameters on the transverse axis in connection with diameter programming can be controlled selectively with the machine data:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK (definition of tool parameters).

Details are described with the mentioned MD

#### DRF handwheel traversal with half distance

During DRF handwheel traversal, it is possible to move a transverse axis through only half the distance of the specified increment as follows:

Specify the distance with handwheel via the machine data:

MD11346 $MN_HANDWHEEL_TRUE_DISTANCE = 1 (handwheel path or speed specification)

Define the DRF offset in the transverse axis as a diameter offset with the machine data:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK bit 9 = 1 (definition of tool parameters)

Deselecting an axial DRF compensation (DRFOF) also deletes an existing tool compensation (handwheel override in tool direction).

**Note**
For further information about superimposed movements with the handwheel, please refer to:

**References:**
Function Manual, Extended Functions; Manual and Handwheel Travel (H1)

Programming Manual, Fundamentals

(The Programming Manual describes the complete technical program options in order to deselect the DRF offset axis-specifically.)
18.3 Flat D number structure

18.3.1 General

Simple tool management

Simple tool management (no replacement tools, no magazines) using D numbers is possible for turning machines. The function is available in the basic level of tool management (without tool management function activated). Grinding tools cannot be defined using this function.

Activation

Which type of D number management is valid may be set via the machine data:

MD18102 $MN_MN_TYPE_OF_CUTTING_EDGE (type of D number programming).

<table>
<thead>
<tr>
<th>Value</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>As previously = default setting</td>
</tr>
<tr>
<td>1</td>
<td>Flat D number structure with absolute direct D programming</td>
</tr>
</tbody>
</table>

Cutting edges can be deleted individually via PI command or NC programming command. Cutting edges with a specific number can also be created selectively using HMI.

18.3.2 Creating a new D number (compensation block)

Programming

Tool compensations can be programmed with system variables $TC_DP1 to $TC_DP25. The contents have the same meaning as before.

The syntax changes: no T number is specified.

- "Flat D number" function active:
  
  $TC_DP[x][d] = $value ; where x=parameter no., d=D number

  i.e., data with this syntax can only be loaded to the NCK if the "Flat D number" function is activated.

- "Flat D number" function inactive:
  
  $TC_DP[x][t][d] = $value ; where t=T number, d=D number

A D number can only be assigned once for each tool, i.e., each D number stands for exactly one compensation data block.

A new data block is stored in the NCK memory when a D number that does not exist is created for the first time.

The maximum number of D or offset data blocks (max. 600) is set via the machine data:
MD18100 $MN_MM_NUM_CUTTING_EDGES_IN_TOA (tool compensations in TO area).

Data backup

Data backup is carried out in the same format, i.e., a backup file created with the "Flat D number" function cannot be loaded on the NCK of a control that has not activated the function. This also applies in reverse for transfer.

D range

1 - 99 999 999

18.3.3 D number programming

MD18102 = 1

If MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE = 1, then D compensation is activated without reference to a certain tool. 

$D0 still contains the previous significance, "Deselection of active compensation in NCK".

Address extension of D

It is not possible to extend the address of D. Only one active compensation data block is possible for the tool path at a given time.

Programming

Programming in the part program is carried out as before. Only the value range of the programmed D number is increased.

Example 1:

<table>
<thead>
<tr>
<th>MD parameterization</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD22550 $MC_TOOL_CHANGE_MODE</td>
<td>= 0 Tool change with programming of T.</td>
</tr>
<tr>
<td>MD18102 $MN_MM_TYPE_OF_CUTTING_EDGE</td>
<td>= 1 D number programming without reference to a certain tool.</td>
</tr>
<tr>
<td>MD20270 $MC_CUTTING_EDGE_DEFAULT</td>
<td>= -1 D compensation remains unchanged if tool is changed.</td>
</tr>
</tbody>
</table>

Program code

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>D92</td>
<td></td>
</tr>
<tr>
<td>X0</td>
<td>; Traverse with compensations from D92.</td>
</tr>
<tr>
<td>T17</td>
<td>; Outputs T=17 to the PLC</td>
</tr>
<tr>
<td>X1</td>
<td>; Traverse with compensations from D92.</td>
</tr>
</tbody>
</table>
Program code | Comment
---|---
D16 X2 | Traverse with compensations from D16.
D32000 X3 | Traverse with compensations from D32000.
T29000 X4 | Outputs T=29000 to the PLC.
D1 X5 | Traverse with compensations from D1.

Example 2:

MD22550 = 0

Program code | Comment
---|---
T1 T2 T3 | No waiting, D777 is activated, T3= programmed and active tool in the display, D777= programmed and active compensation.

Note

The tool change and the assignment of a D compensation to an actual tool are the responsibility of the NC program and of the PLC program, if applicable.

Delete D no. via part program

- **With** flat D number:
  
  `$TC_DP1[d] = 0`
  
  Compensation data block with the number D in the TO unit is deleted. The memory is then free for the definition of another D number.

- **Without** flat D number:
  
  `$TC_DP1[t][d] = 0`
  
  Cutting edge d of tool t is deleted.

- `$TC_DP1[0] = 0`
  
  All D compensations of the TO unit are deleted.

Active compensation data blocks (D numbers) cannot be deleted. This means, that it may be necessary to program D0 before deleting.
Tool MDs

The following machine data affect the way tools and cutting edges (D numbers) work in the NCK:

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD20270 $MC_CUTTING_EDGE_DEFAULT</td>
<td>Standard tool cutting edge after tool change</td>
</tr>
<tr>
<td>MD20130 $MC_CUTTING_EDGE_RESET_VALUE</td>
<td>Tool cutting edge - Length compensation on power-up (RESET / TP end)</td>
</tr>
<tr>
<td>MD20120 $MC_TOOL_RESET_VALUE</td>
<td>Tool - Length compensation on power-up (RESET / TP end)</td>
</tr>
<tr>
<td>MD20121 $MC_TOOL_PRESEL_RESET_VALUE</td>
<td>Preselect tool on RESET</td>
</tr>
<tr>
<td>MD22550 $MC_TOOL_CHANGE_MODE</td>
<td>Tool change with M function instead of T function</td>
</tr>
<tr>
<td>MD22560 $MC_TOOL_CHANGE_M_CODE</td>
<td>M function for tool change</td>
</tr>
<tr>
<td>MD20110 $MC_RESET_MODE_MASK</td>
<td>Definition of basic control settings after RESET/TP-End)</td>
</tr>
<tr>
<td>MD20112 $MC_START_MODE_MASK</td>
<td>Determination of basic control settings after NC start</td>
</tr>
</tbody>
</table>

18.3.4 Programming the T number

When the "Flat D number structure" function is active, NC address T continues to be evaluated, i.e., the programmed T number and the active T number are displayed. However, the NC determines the D number without reference to the programmed T value.

The NC detects 1 master spindle per channel (via the spindle number, which can be set using MD). Compensations and the \textit{M6} command (tool change) are only calculated in reference to the master spindle.

An address extension T is interpreted as a spindle number (e.g., T2 = 1; tool 1 to be selected on spindle 2); a tool change is only detected if spindle 2 is the master spindle.

18.3.5 Programming M6

MD22550 and MD22560

The NC detects 1 master spindle per channel (via the spindle number, which can be set using MD). Compensations and the \textit{M6} command (tool change) are only calculated in reference to the master spindle.

Whether the tool change command is performed with an M function is defined via the machine data:

MD22550 $MC_TOOL_CHANGE_MODE (new tool compensation with M function).

T is used as the tool preparation command.

The name of the M function for the tool change is defined via the machine data:
**Example**

Two spindles are defined, spindle 1 and spindle 2, and the following applies:

| MD20090 = 2 | ; Spindle no. 2 is the master spindle. |
| M6           | ; Tool change desired, command refers implicitly to the master spindle. |
| M1 = 6       | ; No tool change, since spindle no. 2 is the master spindle. |
| M2 = 6       | ; Tool is changed, since spindle no. 2 is the master spindle. |

**18.3.6 Program test**

**MD20110**

To have the active tool and the tool compensation included as follows, can be defined via the machine data:

MD20110 $MC_RESET_MODE_MASK, Bit 3 (Definition of control default settings after RESET/TP end).

<table>
<thead>
<tr>
<th>Value</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 3</td>
<td>= 1 From the last test program to finish in test mode</td>
</tr>
<tr>
<td></td>
<td>= 0 From the last program to finish before activation of the program test</td>
</tr>
</tbody>
</table>

**Prerequisite**

The bits 0 and 6 must be set by the machine data:

MD20110 $MC_RESET_MODE_MASK, Bit 3 (Definition of control default settings after RESET/TP end).
18.3.7 Tool management or "Flat D numbers"

Tool management

NCK active tool management works on the basis of the following assumptions:

1. Tools are managed in magazines.
2. Cutting edges are monitored; limits reached cause the tool to be disabled.
3. Idea of the sister tools: Tools are programmed for selection only on the basis of their name. NCK then selects the concrete tool according to the strategy.

Note

For SINUMERIK 828D, this function is only available as an option!

This means that it only makes sense to employ tool management when specific tools have been defined and these are to be utilized by the NCK.

Flat D number

Flat D number means that tool management is carried out outside the NCK and there is no reference made to T numbers.

No mixture of tool management and flat D number

It does not make sense to mix or distribute the tool management functions over the NCK and PLC, since the main reason for tool management on the NCK is to save time.

This only works if the tasks that are time-critical are carried out on the NCK. This is not the case for "Flat D number", however.

Note

Activation of both of the functions "Flat D number structure" and "Tool management" is monitored. If both are activated at the same time, "Tool management" takes priority.
18.4 Tool cutting edge

18.4.1 General

Tool cutting edge

The following data are used to describe a tool cutting edge uniquely:

- Tool type (end mill, drill, etc.)
- Geometrical description
- Technological description

Tool parameter

The geometrical description, the technological description and the tool type are mapped to tool parameters for each tool cutting edge.

The following tool parameters are available for the relevant tool types:

<table>
<thead>
<tr>
<th>Tool parameter</th>
<th>Significance</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tool type</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Cutting edge position</td>
<td>for turning tools or for milling/grinding tools</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with 2D TRC contour tool</td>
</tr>
<tr>
<td>3</td>
<td>Length 1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Length 2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Length 3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Radius 1/Length 1</td>
<td>for 3D face milling</td>
</tr>
<tr>
<td>7</td>
<td>Length 2</td>
<td>for 3D face milling</td>
</tr>
<tr>
<td>8</td>
<td>Radius 1</td>
<td>for 3D face milling</td>
</tr>
<tr>
<td>9</td>
<td>Radius 2</td>
<td>for 3D face milling</td>
</tr>
<tr>
<td>10</td>
<td>Angle 1 / minimum threshold angle</td>
<td>for 3D face milling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with 2D TRC contour tool</td>
</tr>
<tr>
<td>11</td>
<td>Angle 2 / minimum threshold angle</td>
<td>for 3D face milling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with 2D TRC contour tool</td>
</tr>
<tr>
<td>12</td>
<td>Length 1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Length 2</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Length 3</td>
<td></td>
</tr>
</tbody>
</table>
### Tool parameter 1: Tool type

**Description**

The tool type (3digit number) defines the tool in question. The selection of this tool type determines further components such as geometry, wear and tool base dimensions in advance.

**3D-face milling**

Milling cutter types 111, 120, 121, 130, 155, 156 and 157 are given special treatment for 3D-face milling by evaluating tool parameters (1 -23).

**References**

For more information about various tool types, see:

- Functions Manual - Basic Functions; Tool Offset (W1), Chapter "Tool type (tool parameters)"
- Programming Manual Fundamentals; Chapter: "Tool compensations" > "List of tool types"
- Functions Manual - Special Functions; 3D-tool radius compensation (W5)

<table>
<thead>
<tr>
<th>Tool parameter</th>
<th>Significance</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Radius 1/Length 1</td>
<td>for 3D face milling</td>
</tr>
<tr>
<td>16</td>
<td>Length 2</td>
<td>for 3D face milling</td>
</tr>
<tr>
<td>17</td>
<td>Radius 1</td>
<td>for 3D face milling</td>
</tr>
<tr>
<td>18</td>
<td>Radius 2</td>
<td>for 3D face milling</td>
</tr>
<tr>
<td>19</td>
<td>Angle 1 / minimum limit angle</td>
<td>for 3D face milling with 2D TRC contour tool.</td>
</tr>
<tr>
<td>20</td>
<td>Angle 2 / maximum limit angle</td>
<td>for 3D face milling with 2D TRC contour tool.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tool parameter</th>
<th>Significance</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>Basic length 1</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Basic length 2</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Basic length 3</td>
<td></td>
</tr>
</tbody>
</table>

* "Reserved" means that this tool parameter is not used and is reserved for expansions.
Conditions

The following is applicable to the "Tool type" parameter:

- The tool type must be specified for each tool cutting edge.
- Only the values specified can be used for the tool type.
- Tool type "0" (zero) means that no valid tool has been defined.
### Tool types and tool parameters

Different tool types and the most important tool parameters are listed in the following table. The tool parameters available for a certain tool type are designated with "x".

<table>
<thead>
<tr>
<th>Tool type</th>
<th>Tool parameter $STC_DP</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milling tools and drilling tools (all others)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>Milling tool according to CLDATA$^1$</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>110</td>
<td>Ball end mill</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>120</td>
<td>End mill</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>121</td>
<td>End mill with corner rounding</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>130</td>
<td>Angle head cutter</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>131</td>
<td>Angle head mill with corner rounding</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>140</td>
<td>Facing cutter</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>145</td>
<td>Thread cutter</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>150</td>
<td>Side mill</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>155</td>
<td>Bevel cutter</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>200</td>
<td>Twist drill</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>205</td>
<td>Solid drill</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>210</td>
<td>Boring bar</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>220</td>
<td>Center drill</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>230</td>
<td>Countersink</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>231</td>
<td>Counterbore</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>240</td>
<td>Tap regular thread</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>241</td>
<td>Tap fine thread</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>242</td>
<td>Whitworth thread tap</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>250</td>
<td>Reamer</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Grinding tools and turning tools (400 – 599)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>Surface grinding wheel</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>401</td>
<td>Surface grinding wheel with monitoring</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>403</td>
<td>Same as 401 but w/o tool base dimension for SUG$^2$</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>410</td>
<td>Facing wheel</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>411</td>
<td>Facing wheel with monitoring</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>413</td>
<td>Same as 411 but w/o tool base dimension for SUG$^2$</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>490</td>
<td>Dresser</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>500</td>
<td>Roughing tool</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>510</td>
<td>Finishing tool</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>520</td>
<td>Plug cutter</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>530</td>
<td>Parting tool</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>540</td>
<td>Threading tool</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>550</td>
<td>Button tool / forming tool (TOOLMAN)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Special tools (700)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>700</td>
<td>Slotting saw</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

---

1. CLDATA$^1$ “cutter location data” — Cutter location data in accordance with DIN 66215, BI 1 SUG$^2$ — Grinding wheel peripheral speed
**W1: Tool offset**

18.4 Tool cutting edge

---

**Note**

The tool type has no significance in the turning tool groups.

Nonlisted numbers are also permitted, in particular with grinding tools (400-499).

---

**Tool offset data**

Tool offset data (TOA data) is stored in the system variables.

Example Slotting saw tool type (Type 700)

<table>
<thead>
<tr>
<th></th>
<th>Geometry</th>
<th>Wear</th>
<th>Base</th>
<th>Einheit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Length compensation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Length 1</td>
<td>$TC_DP3</td>
<td>$TC_DP12</td>
<td>$TC_DP21</td>
<td>mm</td>
</tr>
<tr>
<td>Length 2</td>
<td>$TC_DP4</td>
<td>$TC_DP13</td>
<td>$TC_DP22</td>
<td>mm</td>
</tr>
<tr>
<td>Length 3</td>
<td>$TC_DP5</td>
<td>$TC_DP14</td>
<td>$TC_DP23</td>
<td>mm</td>
</tr>
<tr>
<td><strong>Saw blade compensation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diameter d</td>
<td>$TC_DP6</td>
<td>$TC_DP15</td>
<td></td>
<td>mm</td>
</tr>
<tr>
<td>Slot width b</td>
<td>$TC_DP7</td>
<td>$TC_DP16</td>
<td></td>
<td>mm</td>
</tr>
<tr>
<td>Projection k</td>
<td>$TC_DP8</td>
<td>$TC_DP17</td>
<td></td>
<td>mm</td>
</tr>
</tbody>
</table>
Figure 18-3  Geometry of slotting saw (analogous to angle head cutter)

The width of the saw blade is accounted for with tool radius compensation (G40 to G42 as follows:

<table>
<thead>
<tr>
<th>Command</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>G40</td>
<td>No saw blade compensation</td>
</tr>
<tr>
<td>G41</td>
<td>Saw blade compensation left</td>
</tr>
<tr>
<td>G42</td>
<td>Saw blade compensation right</td>
</tr>
</tbody>
</table>

18.4.3 Tool parameter 2: Cutting edge position

Description

The cutting edge position describes the position of the tool tip P in relation to the cutting edge center point S. It is entered in tool parameter 2.

The cutting edge position is required together with the cutting edge radius (tool parameter 8) for the calculation of the tool radius compensation for turning tools (tool type 5xx).
18.4 Tool cutting edge

Figure 18-4  Dimensions for turning tools: Turning tool

Cutting edge position parameter values

Figure 18-5  Tool parameter 2 (P2): Machining behind the turning center

Figure 18-6  Tool parameter 2 (P2): Machining in front of the turning center
Special points to be noted

- If the cutting edge center point S is used instead of point P as a reference point to calculate the tool length compensation, the identifier 9 must be entered for the cutting edge position.
- The identifier 0 (zero) is not permitted as a cutting edge position.

18.4.4 Tool parameters 3 - 5: Geometry - tool lengths

Description

The lengths of the tools are required for the geometry tool length compensation. They are input as tool lengths 1 to 3 in the tool parameters 3 to 5. The following length specifications must be entered as a minimum for each tool type:

<table>
<thead>
<tr>
<th>Tool type</th>
<th>Required tool lengths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool type 12x, 140, 145, 150:</td>
<td>Tool length 1</td>
</tr>
<tr>
<td>Tool type 13x:</td>
<td>Tool length 1 to 3 (depending on plane G17-G19)</td>
</tr>
<tr>
<td>Tool type 2xx:</td>
<td>Tool length 1</td>
</tr>
<tr>
<td>Tool type 5xx:</td>
<td>Tool length 1 to 3</td>
</tr>
</tbody>
</table>

Example Twist drill (tool type 200) with tool length (tool parameter 3)
Note

All three tool parameters 3 to 5 (tool length 1 to 3) are always calculated in the three geometry axes, irrespective of the tool type.

If more tool lengths are input in the tool parameters 3 to 5 for a tool type than is required as the minimum, then these extra tool lengths are settled in the geometry axes without any alarm.

Special points to be noted

The active size of the tool is only defined when the geometry tool length compensation (tool parameters 3 to 5) and the wear tool length compensation (tool parameters 12 to 14) are added together. The base-dimension/adapter-dimension tool length compensation is also added in order to calculate the total tool length compensation in the geometry axes.

References

For information about entering tool dimensions (lengths) in tool parameters 3 to 5 (tool lengths 1 to 3) and how these are calculated in the three geometry axes, please refer to → Operating Manual.

18.4.5 Tool parameters 6 - 11: Geometry - tool shape

Meaning

The shape of the tool is defined using the tool parameters 6 to 11. The data is required for the geometry tool radius compensation.
In most cases, only tool parameter 6 (tool radius 1) is used.

<table>
<thead>
<tr>
<th>Tool parameter</th>
<th>Meaning</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Tool length 1</td>
<td>Not used</td>
</tr>
<tr>
<td>7</td>
<td>Tool length 2</td>
<td>Not used</td>
</tr>
<tr>
<td>8</td>
<td>Tool radius 1</td>
<td>The tool radius must be entered for the following tool types in tool parameter 6 (tool radius 1):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tool type 1xx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tool type 5xx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A tool radius does not have to be entered for drilling tools (tool type 2xx).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The cutting edge position (tool parameter 2) also has to be entered for turning tools (tool type 5xx).</td>
</tr>
<tr>
<td>9</td>
<td>Tool radius 2</td>
<td>Not used</td>
</tr>
<tr>
<td>10</td>
<td>Tool angle 1</td>
<td>Not used</td>
</tr>
<tr>
<td>11</td>
<td>Tool angle 2</td>
<td>Not used</td>
</tr>
</tbody>
</table>

2D TRC with contour tools

For the definition of contour tools with multiple tool cutting edges, the minimum and maximum limit angle can be entered. Both limit angles each relate to the vector of the cutting edge center point to the cutting edge reference point and are counted clockwise.

<table>
<thead>
<tr>
<th>Tool angle 1</th>
<th>Minimum limit angle per tool cutting edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool angle 2</td>
<td>Maximum limit angle per tool cutting edge</td>
</tr>
</tbody>
</table>

3D face milling

Tool parameters 6 to 11 are required for tool description of 3D face milling.
References

Please refer to the following documentation for information about entering tool shapes (radius for tool radius compensation) in tool parameters 6 to 11 and how these are calculated by geometry tool radius compensation in the three geometry axes:

- Programming Manual, Fundamentals; Section: "Tool compensations" > "2½ D tool compensation"
- Functions Manual - Special Functions; 3D tool radius compensation (W5)

For 3D face milling, please refer to:

- Programming Manual, Job Planning; Section: "Transformations" > "Three, Four and Five axis Transformation (TRAORI)"

18.4.6 Tool parameters 12 - 14: Wear - tool lengths

Description

While geometry tool length compensation (tool parameters 3 to 5) is used to define the size of the tool, wear tool length compensation can be used to correct the change in the active tool size.

The active tool dimensions can change due to:

- Differences between the tool fixture on the tool measurement machine and the tool fixture on the machine tool
- Tool wear caused during service life by machining
- Definition of the finishing allowances

Active tool size

The geometry tool compensation (tool parameters 3 to 5) and the wear tool length compensation (tool parameters 12 to 14) are added together (geometry tool length 1 is added to wear tool length 1, etc.) to arrive at the size of the active tool.

18.4.7 Tool parameters 15 - 20: Wear - tool shape

Description

While geometry tool radius compensation (tool parameters 6 to 11) is used to define the shape of the tool, wear tool radius compensation can be used to correct the change in the active tool shape.

The active tool dimensions can change due to:

- Tool wear caused during service life by machining
- Definition of the finishing allowances
Active tool shape

The geometry tool radius compensation (tool parameters 6 to 11) and the wear tool radius compensation (tool parameters 15 to 20) are added together (geometry tool radius 1 is added to wear tool radius 1, etc.) to arrive at the shape of the active tool.

18.4.8 Tool parameters 21 - 23: Tool base dimension/adapter dimension

Description

Tool base dimension/adapter dimension can be used when the reference point of the toolholder (tool size) differs from the reference point of the toolholder.

This is the case when:

- The tool and the tool adapter are measured separately but are installed on the machine in one unit (the tool size and adapter size are entered separately in a cutting edge).
- The tool is used in a second tool fixture located in another position (e.g. vertical and horizontal spindle).
- The tool fixtures of a tool turret are located at different positions.

Figure 18-8 Application examples for base-dimension/adapter-dimension TLC
Tool basic length 1 to 3 (tool parameters 21 to 23)

In order that the discrepancy between the toolholder reference point F and the toolholder reference point F’ can be corrected on the three geometry axes (three dimensional), all 3 basic lengths are active irrespective of the tool type. In other words, a twist drill (tool type 200) with a tool length compensation (length 1) can also have a tool base dimension/adapter dimension in 3 axes.

References

Please refer to the following documentation for more information about base-dimension/adapter-dimension tool length compensation:

- Programming Manual, Fundamentals

18.4.9 Tool parameter 24: Undercut angle

Meaning

Certain turning cycles, in which traversing motions with tool clearance are generated, monitor the tool clearance angle of the active tool for possible contour violations.

Value range

The angle (0 to 90° with no leading sign) is entered in tool parameter 24 as the tool clearance angle.

Figure 18-9 Tool clearance angle of the turning tool during relief cutting
**Machining type, longitudinal or transverse**

The tool clearance angle is entered in different ways according to the type of machining (longitudinal or face). If a tool is to be used for both longitudinal and face machining, two cutting edges must be entered for different tool clearance angles.

![Diagram of tool clearance angle for longitudinal and face machining]

**Figure 18-10 Tool clearance angle for longitudinal and face machining**

**Note**

If a tool clearance angle (tool parameter 24) of zero is entered, relief cutting is not monitored in the turning cycles.

**References**

Please refer to the following documentation for a detailed description of the tool clearance angle:

- Programming Manual Cycles
18.4.10 Tools with a relevant tool point direction

The following must be observed for tools with relevant cutting edge position:

- The straight line between the tool edge center points at the block start and block end is used to calculate intersection points with the approach and retraction block. The difference between the tool edge reference point and the tool edge center point is superimposed on this movement. For approach and/or retraction with \texttt{KONT}, the movement is superimposed in the linear subblock of the approach or retraction movement. Therefore, the geometric conditions for tools with or without relevant cutting edge position are identical.

- In circle blocks and in motion blocks containing rational polynomials with a denominator degree $> 4$, it is not permitted to change a tool with active tool radius compensation in cases where the distance between the tool edge center point and the tool edge reference point changes. With other types of interpolation, it is now possible to change when a transformation is active (e.g. \texttt{TRANSMIT}).

- For tool radius compensation with variable tool orientation, the transformation from the tool edge reference point to the tool edge center point can no longer be performed by means of a simple zero offset. Tools with a relevant cutting edge position are therefore not permitted for 3D peripheral milling (an alarm is output).

\textbf{Note}

The subject is irrelevant with respect to face milling as only defined tool types without relevant cutting edge position are permitted for this operation anyway. (A tool with a type, which has not been explicitly approved, is treated as a ball end mill with the specified radius. A cutting edge position parameter is ignored).

18.5 Tool radius compensation 2D (TRC)

18.5.1 General

\textbf{Note}

For tool radius compensation (TRC) see:

\textbf{References:}

Programming Manual Fundamentals

Only the Programming Guide contains a complete technical description of the tool radius compensation (TRC) and its special aspects.
Why TRC?

The contour (geometry) of the workpiece programmed in the part program should be independent of the tools used in production. This makes it necessary to draw the values for the tool length and tool radius from a current offset memory. Tool radius compensation can be used to calculate the equidistant path to the programmed contour from the current tool radius.

![Workpiece contour (geometry) with equidistant path](image)

Figure 18-11 Workpiece contour (geometry) with equidistant path

TRC on the plane

TRC is active on the current plane (G17 to G19) for the following types of interpolation:

<table>
<thead>
<tr>
<th>Type of Interpolation</th>
<th>Command</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear interpolation</td>
<td>G0, G1</td>
<td></td>
</tr>
<tr>
<td>Circular interpolation</td>
<td>G2, G3, CIP</td>
<td></td>
</tr>
<tr>
<td>Helical interpolation</td>
<td>G2, G3</td>
<td></td>
</tr>
<tr>
<td>Spline interpolation</td>
<td>ASPLINE, BSPLINE, CSPLINE</td>
<td></td>
</tr>
<tr>
<td>Polynomial interpolation</td>
<td>POLY</td>
<td></td>
</tr>
</tbody>
</table>

18.5.2 Selecting the TRC (G41/G42)

Direction of compensation

TRC calculates a path, which is equidistant to the programmed contour. Compensation can be performed on the left- or righthand side of the programmed contour in the direction of motion.

<table>
<thead>
<tr>
<th>Command</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>G41</td>
<td>TRC on the lefthand side of the contour in the direction of motion</td>
</tr>
<tr>
<td>G42</td>
<td>TRC on the righthand side of the contour in the direction of motion</td>
</tr>
<tr>
<td>G40</td>
<td>Deselection of TRC</td>
</tr>
</tbody>
</table>
Intermediate blocks

In general, only program blocks with positions on geometry axes in the current plane are programmed when TRC is active. However, dummy blocks can still also be programmed with active TRC. Dummy blocks are program blocks, which do not contain any positions on a geometry axis in the current plane:

- Positions on the infeed axis
- Auxiliary functions,
- etc.

The maximum number of dummy blocks can be defined in the machine data:

MD20250 $MC_CUTCOM_MAXNUM_DUMMY_BLOCKS (Max no. of dummy blocks with no traversing movements for TRC).

Special points to be noted

- TRC can only be selected in a program block with G0 (rapid traverse) or G1 (linear interpolation).
- A tool must be loaded (T function) and the tool cutting edge (tool compensation) (D1 to D9) activated no later than in the program block with the tool radius compensation selection.
- Tool radius compensation is not selected with a tool cutting edge/tool compensation of D0.
- If only one geometry axis is programmed on the plane when tool radius compensation is selected, the second axis is automatically added on the plane (last programmed position).
- If no geometry axis is programmed for the current plane in the block with the tool radius compensation selection, no selection takes place.
- If tool radius compensation is deselected (G40) in the block following tool radius compensation selection, no selection takes place.
- If tool radius compensation is selected, the approach behavior is determined by the NORM/KONT instructions.

18.5.3 Approach and retraction behavior (NORM/KONT/KONTC/KONTT)

NORM and KONT

The NORM and KONT instructions can be used to control approach behavior (selection of tool radius compensation with G41/42) and retraction behavior (deselection of tool radius compensation with G40):

<table>
<thead>
<tr>
<th>Command</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORM</td>
<td>Normal setting at start point/end point (initial setting)</td>
</tr>
<tr>
<td>KONT</td>
<td>Follow contour at start point/end point</td>
</tr>
<tr>
<td>KONTC</td>
<td>Approach/retraction with constant curvature</td>
</tr>
<tr>
<td>KONTT</td>
<td>Approach/retraction with constant tangent</td>
</tr>
</tbody>
</table>
Special points to be noted

- **KONT** only differs from **NORM** when the tool start position is behind the contour.

![Diagram showing tool path](image)

Figure 18-12 Example for selecting TRC with **KONT** or **NORM** in front of and behind the contour

- **KONT** and **G450/G451** (corner behavior at outer corners) has a general effect and determines the approach and retraction behavior with TRC.

- When tool radius compensation is deselected, the retraction behavior is determined by the **NORM/KONT** instructions.

Supplementary conditions

The approach and retraction blocks are polynomials in the following two variants. Therefore, they are only available for control variants, which support polynomial interpolation.

- **KONTT**
  With **KONTT**, approach and retraction to/from the contour is with a constant tangent. The curvature at the block transition is not usually constant.

- **KONTC**
  With **KONTC**, not only the tangent but also the curvature is constant at the transition, with the result that a jump in acceleration can no longer occur on activation/deactivation.

Although **KONTC** includes the **KONTT** property, the constant tangent version **KONTT** is available on its own, because the constant curvature required by **KONTC** can produce undesired contours.
Axes

The continuity condition is observed in all three axes. It is thus possible to program a simultaneous path component perpendicular to the compensation plane for approach/retraction.

Only linear blocks are permitted for the original approach and retraction blocks with KONTT/KONTC. These programmed linear blocks are replaced in the control by the corresponding polynomial curves.

Exception

KONTT and KONTC are not available in 3D variants of tool radius compensation (CUT3DC, CUT3DCC, CUT3DF).

If they are programmed, the control switches internally to NORM without an error message.

Example for KONTC

The two figures below show a typical application for approach and retraction with constant curvature:

The full circle is approached beginning at the circle center point. The direction and curvature radius of the approach circle at the block end point are identical to the values of the next circle. Infeed takes place in the Z direction in both approach/retraction blocks simultaneously.

The associated NC program segment is as follows:

```
$TC_DP1[1,1]=121 ; Milling tool
$TC_DP6 [1,1]=10 ; Radius 10 mm
N10 G1 X0 Y0 Z60 G64 T1 D1 F10000
N20 G41 KONTC X70 Y0 Z0
N30 G2 I=70 ; Full circle
N40 G40 G1 X0 Y0 Z60
N50 M30
```

Explanation:

In this example, a full circle with a radius of 70 mm is machined in the X/Y plane. Since the tool has a radius of 10 mm, the resulting tool center point path describes a circle with a radius of 60 mm. The start/end points are at X0 Y0 Z60, with the result that a movement takes place in the Z direction at the same time as the approach/retraction movement in the compensation plane.
Figure 18-13  Approach and retraction with constant curvature during inside machining of a full circle: Projection in the X-Y plane.
KONTT and KONTC compared

The figure below shows the differences in approach/retraction behavior between KONTT and KONTC. A circle with a radius of 20 mm about the center point at X0 Y-40 is compensated with a tool with an external radius of 20 mm. The tool center point therefore moves along a circular path with radius 40 mm. The end point of the approach blocks is at X40 Y30. The transition between the circular block and the retraction block is at the zero point. Due to the extended continuity of curvature associated with KONTC, the retraction block first executes a movement with a negative Y component. This will often be undesired. This response does not occur with the KONTT retraction block. However, with this block, an acceleration step change occurs at the block transition.

If the KONTT or KONTC block is the approach block rather than the retraction block, the contour is exactly the same, but is simply machined in the opposite direction, i.e. the approach and retraction behavior are symmetrical.
18.5.4 Smooth approach and retraction

18.5.4.1 Function

Description

The SAR (Smooth Approach and Retraction) function is used to achieve a tangential approach to the start point of a contour, regardless of the position of the start point.

The approach behavior can be varied and adapted to special needs using a range of additional parameters.

The two functions, smooth approach and smooth retraction, are largely symmetrical. The following section is, therefore, restricted to a detailed description of approach; special reference is made to differences affecting retraction.
Sub-movements

There are maximum 4 sub-movements in case of soft retraction and approach with the following positions:

- Start point of the movement $P_0$
- Intermediate points $P_1$, $P_2$ and $P_3$
- End point $P_4$

Points $P_0$, $P_3$ and $P_4$ are always defined. Intermediate points $P_1$ and $P_2$ can be omitted, according to the parameters defined and the geometrical conditions.

On retraction, the points are traversed in the reverse direction, i.e. starting at $P_4$ and ending at $P_0$.

18.5.4.2 Parameters

The response of the smooth approach and retraction function is determined by up to 9 parameters:

Non-modal G code for defining the approach and retraction contour

This G code cannot be omitted.

- $G147$: Approach with a straight line
- $G148$: Retraction with a straight line
- $G247$: Approach with a quadrant
- $G248$: Retraction with a quadrant
- $G347$: Approach with a semicircle
- $G348$: Retraction with a semicircle

![Approach behavior depending on $G147$ to $G347$ and DISR (with simultaneous activation of tool radius compensation)](image-url)
Modal G code for defining the approach and retraction contour

This G code is only relevant if the approach contour is a quadrant or semicircle. The approach and retraction direction can be determined as follows:

- **G140:**
  Defining the approach and retraction direction using active tool radius compensation. (G140 is the initial setting.)
  
  With positive tool radius:
  - G41 active → approach from left
  - G42 active → approach from right

  If no tool radius compensation is active (G40), the response is identical to G143. In this case, an alarm is not output. If the radius of the active tool is 0, the approach and retraction side is determined as if the tool radius were positive.

- **G141:**
  Approach contour from left, or retract to the left.

- **G142:**
  Approach contour from right, or retract to the right.

- **G143:**
  Automatic determination of the approach direction, i.e., the contour is approached from the side where the start point is located, relative to the tangent at the start point of the following block (P4).

**Note**

The tangent at the end point of the preceding block is used accordingly on retraction. If the end point is not programmed explicitly on retraction, i.e., if it is to be determined implicitly, G143 is not permitted on retraction, since there is a mutual dependency between the approach side and the position of the end point. If G143 is programmed in this case, an alarm is output. The same applies if, when G140 is active, an automatic switchover to G143 takes place as a result of an inactive tool radius compensation.
Modal G code (G340, G341), which defines the subdivision of the movement into individual blocks from the start point to the end point.

**Figure 18-17** Sequence of the approach movement depending on G340/G341

**G340:** The approach characteristic from P₀ to P₄ is shown in the figure.
- If G247 or G347 is active (quadrant or semicircle) and start point P₃ is outside the machining plane defined by the end point P₄, a helix is inserted instead of a circle.
- Point P₂ is not defined or coincides with P₃.
- The circle plane or the helix axis is determined by the plane, which is active in the SAR block (G17-G19), i.e., the projection of the start tangent is used by the following block, instead of the tangent itself, to define the circle.
- The movement from point P₀ to point P₃ takes place along two straight lines at the velocity valid before the SAR block.

**G341:** The approach characteristic from P₀ to P₄ is shown in the figure.
- P₃ and P₄ are located within the machining plane, with the result that a circle is always inserted instead of a helix with G247 or G347.

**Note**
Active, rotating frames are included in all cases where the position of the active plane G17-G19 (circle plane, helix axis, infeed movements perpendicular to the active plane) is relevant.

**DISR**
**DISR** specifies the length of a straight approach line or the radius of an approach arc.

**Retraction/approach with straight lines**
On approach/retraction along a straight line, **DISR** specifies the distance from the cutter edge to the start point of the contour, i.e., the length of the straight line with active TRC is calculated as the total of the tool radius and the programmed value of **DISR**.
An alarm is displayed:

- If DISR is negative and the amount is greater than the tool radius (the length of the resulting approach line is less than or equal to zero).

**Retraction/approach with circles**

Approach/retraction with circles

DISR indicates always the radius of the tool center point path. If tool radius compensation is activated, a circle is generated internally, the radius of which is dimensioned such that the tool center path is derived, in this case also, from the programmed radius.

An alarm is output on approach and retraction with circles:

- If the radius of the circle generated internally is zero or negative
- If DISR is not programmed
- If the radius value ≤ 0.

**DISCL**

DISCL specifies the distance from point P2 from the machining plane.

If the position of point P2 is to be specified by an absolute reference on the axis perpendicular to the circle plane, the value must be programmed in the form DISCL = AC(...).

If DISCL is not programmed, points P1, P2, and P3 are identical with G340 and the approach contour is mapped from P1 to P4.

The system checks that the point defined by DISCL lies between P1 and P3, i.e., in all movements, which have a component perpendicular to the machining plane (e.g., infeed movements, approach movements from P3 to P4), this component must have the same leading sign. It is not permitted to change direction. An alarm is output if this condition is violated.

On detection of a direction reversal, a tolerance is permitted that is defined by the machine data:

MD20204 $MC_WAB_CLEARANCE_TOLERANCE (direction reversal on SAR).

However, if P2 is outside the range defined by P1 and P3 and the deviation is less than or equal to this tolerance, it is assumed that P2 is in the plane defined by P1 and/or P3.

**Example:**

An approach is made with G17 starting at position Z=20 of point P1. The SAR plane defined by P3 is at Z=0. The point defined by DISCL must, therefore, lie between these two points. MD20204=0.010. If P2 is between 20.000 and 20.010 or between 0 and -0.010, it is assumed that the value 20.0 or 0.0 is programmed. The alarm is output if the Z position of P2 is greater than 20.010 or less than -0.010.

Depending on the relative position of start point P0 and end point P4 with reference to the machining plane, the infeed movements are performed in the negative (normal for approach) or positive (normal for retraction) direction, i.e., with G17 it is possible for the Z component of end point P4 to be greater than that of start point P0.

**Programming the end point P4 (or P0 for retraction) generally with X... Y... Z...**

Possible ways of programming the end point P4 for approach
End point $P_4$ can be programmed in the actual SAR block.

$P_4$ can be determined by the end point of the next traversing block.

Further blocks (dummy blocks) can be inserted between the SAR block and the next traversing block without moving the geometry axes.

The end point is deemed to have been programmed in the actual SAR block for approach if at least one geometry axis is programmed on the machining plane (X or Y with \(G17\)). If only the position of the axis perpendicular to the machining plane (Z with \(G17\)) is programmed in the SAR block, this component is taken from the SAR block, but the position in the plane is taken from the following block. In this case, an alarm is output if the axis perpendicular to the machining plane is also programmed in the following block.

**Example:**

\[
\begin{align*}
&T_{\text{DP}1}[1,1]=120 \quad ; \text{Milling tool T1/D1} \\
&T_{\text{DP}6}[1,1]=7 \quad ; \text{Tool with 7 mm radius}
\end{align*}
\]

\[
N10 \quad G90 \quad G0 \quad X0 \quad Y0 \quad Z30 \quad D1 \quad T1
\]

\[
N20 \quad X10
\]

\[
N30 \quad G41 \quad G147 \quad \text{DISCL}=3 \quad \text{DISR}=13 \quad Z=0 \quad \text{F1000}
\]

\[
N40 \quad G1 \quad X40 \quad Y-10
\]

..., ...

\[
N30/N40 \text{ can be replaced by:}
\]

\[
N30 \quad G41 \quad G147 \quad \text{DISCL}=3 \quad \text{DISR}=13 \quad X40 \quad Y-10 \quad Z0 \quad \text{F1000}
\]

or:

\[
N30 \quad G41 \quad G147 \quad \text{DISCL}=3 \quad \text{DISR}=13 \quad \text{F1000}
\]

\[
N40 \quad G1 \quad X40 \quad Y-10 \quad Z0
\]
Possible ways of programming the end point \( P_0 \) for retraction

The end position is always taken from the SAR block, no matter how many axes have been programmed. We distinguish between the following situations:

1. No geometry axis is programmed in the SAR block.
   In this case, the contour ends at point \( P_2 \) (or at point \( P_1 \), if \( P_1 \) and \( P_2 \) coincide). The position in the axes, which describe the machining plane, is determined by the retraction contour (end point of the straight line or arc). The axis component perpendicular to this is defined by DISCL. If, in this case, \( \text{DISCL} = 0 \), the movement takes place completely within the plane.

2. Only the axis perpendicular to the machining plane is programmed in the SAR block.
   In this case, the contour ends at point \( P_1 \). The position of the two other axes is determined in the same way as in 1.

3. At least one axis of the machining plane is programmed.
   The second axis of the machining plane can be determined modally from its last position in the preceding block. The position of the axis perpendicular to the machining plane is generated as described in 1. or 2., depending on whether this axis is programmed or not. The position generated in this way defines the end point \( P_0 \).
   No special measures are required for deselection of tool radius compensation, because the programmed point \( P_0 \) already directly defines the position of the tool center point at the end of the complete contour.
   The start and end points of the SAR contour (\( P_0 \) and \( P_4 \)) can coincide on approach and retraction.

### Retraction with SAR with simultaneous deactivation of TRC

If the SAR retraction block is also used to deactivate tool radius compensation, in the case of 1. and 2., an additional path from \( P_1 \) to \( P_0 \) is inserted such that no movement is produced when tool radius compensation is deactivated at the end of the retraction contour, i.e., this point defines the tool center point and not a position on a contour to be corrected.

### Velocity of the preceding block (typically G0)

All movements from point \( P_1 \) to point \( P_2 \) are performed at this velocity, i.e., the movement parallel to the machining plane and the part of the infeed movement up to the safety clearance.
Programming the feedrate with FAD

<table>
<thead>
<tr>
<th>FAD programmed with ...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>G340</td>
<td>Feedrate from P₂ or P₃ to P₄.</td>
</tr>
<tr>
<td>G341</td>
<td>Feedrate of the infeed movement perpendicular to the machining plane from P₂ to P₃.</td>
</tr>
</tbody>
</table>

If FAD is not programmed, this part of the contour is traversed at the velocity, which is active modally from the preceding block, in the event that no F command defining the velocity is programmed in the SAR block.

Programmed response:

- **FAD=0 or negative** → Alarm Output
- **FAD=...** → Programmed value acts in accordance with the active G code of group 15 (feed type; G93, G94, etc.)
- **FAD=PM(...)** → Programmed value is interpreted as linear feed (like G94), irrespective of the active G code of group 15
- **FAD=PR(...)** → Programmed value is interpreted as revolutionary feed (like G95), irrespective of the active G code of group 15

Example:

```
G0
P0  P1  P2
F000
F2000
F2000

$TC_DP1[1,1]=120 ;Milling tool T1/D1
$TC_DP6[1,1]=7 ;Tool with 7mm radius
N10 G90 G0 X0 Y0 Z20 D1 T1
N20 G41 G341 G247 DISCL=AC(5) DISR=13 FAD 500 X40 Y-10 Z=0 F2000
N30 X50
N40 X60...
```
Programming feed F

This feed value is effective from point P_3 (or from point P_2, if FAD is not programmed). If no F command is programmed in the SAR block, the speed of the preceding block is valid. The velocity defined by FAD is not used for following blocks.

18.5.4.3 Velocities

Velocities at approach

In both approach diagrams below, it is assumed that no new velocity is programmed in the block following the SAR block. If this is not the case, the new velocity comes into effect after point P_4.

<table>
<thead>
<tr>
<th>P0</th>
<th>P1</th>
<th>P2, P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Rapid traverse if G0 is active, otherwise with the old or new F command
- Velocity of preceding block (old F command)
- Infeed velocity programmed with FAD
- New modal velocity programmed with F

Figure 18-18 Velocities in the SAR subblocks on approach with G340
### Velocities at retraction

During retraction, the rolls of the modally active feedrate from the previous block and the programmed feedrate value in the SAR block are interchanged, i.e., the actual retraction contour (straight line, circle, helix) is traversed with the old feedrate value and a new velocity programmed with the F word applies from point P₂ up to P₀.

If even retraction is active and FAD is programmed, the path from P₃ to P₂ is traversed with FAD, otherwise it is traversed with the old velocity. The last F command programmed in a preceding block always applies for the path from P₄ to P₂. G₀ has no effect in these blocks.

Traversing from P₂ to P₀ takes place with the F command programmed in the SAR block or, if no F command is programmed, with the modal F command from a preceding block. This applies on the condition that G₀ is not active.

If rapid traverse is to be used on retraction in the blocks from P₂ to P₀, G₀ must be activated before the SAR block or in the SAR block itself. If an additional F command is programmed in the actual SAR blocks, it is then ineffective. However, it remains modally active for following blocks.

---

**Figure 18-19  Velocities in the SAR subblocks on approach with G341**

<table>
<thead>
<tr>
<th></th>
<th>P0</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No velocity programmed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Only F programmed</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Only FAD programmed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>F and FAD programmed</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Rapid traverse if G0 is active, otherwise with the old or new F command**
- **Velocity of preceding block (old F command)**
- **Infeed velocity programmed with FAD**
- **New modal velocity programmed with F**
18.5.4.4 System variables

Points $P_3$ and $P_4$ can be read in the WCS as system variables during approach.

| $P_{APR}$: | Read $P_3$ (start point) in WCS |
| $P_{AEP}$: | Read $P_4$ (contour start point) in WCS |
| $P_{APDV}$ | $=1$ If the content of $P_{APR}$ and $P_{AEP}$ is valid, i.e., if these contain the position values belonging to the last SAR approach block programmed. |
| | $=0$ The positions of an older SAR approach block are read. |

Changing the WCS between the SAR block and the read operation has no effect on the position values.

18.5.4.5 Supplementary conditions

Supplementary conditions

- Any further NC commands (e.g. auxiliary function outputs, synchronous axis movements, positioning axis movements, etc.) can be programmed in an SAR block. These are executed in the first subblock on approach and in the last subblock on retraction.

- If the end point $P_4$ is not taken from the SAR block but from a subsequent traversing block, the actual SAR contour (straight line, quadrant or semicircle) is traversed in this block. The last subblock of the original SAR block does not then contain traversing information for geometry axes. It is always output, however, because further actions (e.g. single axes) may have to be executed in this block.

Figure 18-20 Velocities in the SAR subblocks on retraction
At least two blocks must always be taken into consideration:

- The SAR block itself
- The block, which defines the approach or retraction direction

Further blocks can be programmed between these two blocks. The number of possible dummy blocks is limited with the machine data: MD20202 $MC_WAB_MAXNUM_DUMMY_BLOCKS (maximum number of blocks with no traversing motions with SAR).

- If tool radius compensation is activated simultaneously in an approach block the first linear block of the SAR contour is the block in which activation takes place. The complete contour generated by the SAR function is treated by tool radius compensation as if it has been programmed explicitly (collision detection, calculation of intersection, approach behavior NORM/KONT).

- The direction of the infeed motion and the position of the circle plane or the helix axis are defined exclusively by the active plane (G17 - G19) - rotated with an active frame where appropriate.

- On approach, a preprocessor stop must not be inserted between the SAR block and the following block which defines the direction of the tangent. Whether programmed explicitly or inserted automatically by the control, a preprocessor stop results, in this case, in an alarm.

Behavior with REPOS

If an SAR cycle is interrupted and repositioned, it resumes at the point of interruption on RMIBL. With RMEBL, the contact point is the end point of the last SAR block; with RMBBL, it is the start point of the first SAR block.

If RMIBL is programmed together with DISPR (reapproach at distance DISPR in front of interruption point), the reapproach point can appear in a subblock of the SAR cycle before the interruption subblock.

18.5.4.6 Examples

Example 1

The following conditions must be true:

- Smooth approach is activated in block N20
- X=40 (end point); Y=0; Z=0
- Approach movement performed with quadrant (G247)
- Approach direction not programmed, G140 is valid, i.e. because TRC is active (G42) and compensation value is positive (10), the contour is approached from the right
- Approach circle generated internally (SAR contour) has radius 20, so that the radius of the tool center path is equal to the programmed value DISR=10
Because of G341, the approach movement takes place with a circle in the plane, resulting in a start point at (20, -20, 0).

Because DISCL=5, point P2 is at position (20, -20, 5) and, because of Z30, point P1 is in N10 at (20, -20, 30).

Figure 18-21 Contour example 1

**Part program:**

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_DP1[1,1]=120 ;</td>
<td>Tool definition T1/D1</td>
</tr>
<tr>
<td>$TC_DP6[1,1]=10 ;</td>
<td>Radius</td>
</tr>
<tr>
<td>N10 G0 X0 Y0 Z30 ;</td>
<td></td>
</tr>
<tr>
<td>N20 G247 G341 G42 NORM D1 T1 Z0 FAD=1000 F=2000 DISCL=5 DISR=10</td>
<td></td>
</tr>
<tr>
<td>N30 X40</td>
<td></td>
</tr>
<tr>
<td>N40 X100</td>
<td></td>
</tr>
<tr>
<td>N50 Y=-30</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Example 2**

The following conditions must be true for approach:

- Smooth approach is activated in block N20
- Approach movement performed with quadrant (G247)
- Approach direction not programmed, G140 is valid, i.e. because TRC is active (G41), the contour is approached from the left
- Contour offset OFFN=5 (N10)
- Current tool radius=10, and so the effective compensation radius for TRC=15; the radius of the SAR contour is thus equal to 25, with the result that the radius of the tool center path is equal to DISR=10
• The end point of the circle is obtained from N30, since only the Z position is programmed in N20

• Infeed motion
  – From Z20 to Z7 (DISCL=AC(7)) with rapid traverse
  – Then on to Z0 with FAD=200
  – Approach circle in X-Y-plane and following blocks with F1500
    (In order that this velocity becomes effective in the following blocks, the active G-code G0 in N30 must be overwritten with G1. Otherwise, the contour would continue to be machined with G0.)

The following conditions must be true for retraction:
• Smooth retraction is activated in block N60
• Retraction movement performed with quadrant (G248) and helix (G340)
• FAD not programmed, since irrelevant for G340
• Z=2 in the start point; Z=8 in the end point, since DISCL=6
• When DISR=5, the radius of SAR contour=20; that of the tool center point path=5
• After the circle block, the retraction movement leads from Z8 to Z20 and the movement is parallel to the XY plane up to the end point at X70 Y0

Figure 18-22  Contour example 2

Part program:

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_DP1[1,1]=120</td>
<td>; Tool definition T1/D1</td>
</tr>
<tr>
<td>$TC_DP6[1,1]=10</td>
<td>; Radius</td>
</tr>
</tbody>
</table>
### Program code

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G0 X0 Y0 Z20 G64 D1 T1 OFFN = 5</td>
<td>; (P0app)</td>
</tr>
<tr>
<td>N20 G41 G247 G341 Z0 DISCL = AC(7) DISR = 10 F1500 FAD=200</td>
<td>; (P3app)</td>
</tr>
<tr>
<td>N30 G1 X30 Y-10</td>
<td>; (P4app)</td>
</tr>
<tr>
<td>N40 X40 Z2</td>
<td></td>
</tr>
<tr>
<td>N50 X50</td>
<td>; (P4ret)</td>
</tr>
<tr>
<td>N60 G248 G340 X70 Y0 Z20 DISCL = 6 DISR = 5 G40 F10000</td>
<td>; (P3ret)</td>
</tr>
<tr>
<td>N70 X80 Y0</td>
<td>; (P0ret)</td>
</tr>
<tr>
<td>N80 M 30</td>
<td></td>
</tr>
</tbody>
</table>

### Note

The contour generated in this way is modified by tool radius compensation, which is activated in the SAR approach block and deactivated in the SAR retraction block.

The tool radius compensation allows for an effective radius of 15, which is the sum of the tool radius (10) and the contour offset (5). The resulting radius of the tool center path in the approach block is therefore 10, and 5 in the retraction block.

### 18.5.5 Deselecting the TRC (G40)

#### G40 instruction

TRC is deselected with the **G40** instruction.

#### Special points to be noted

- **TRC can only be deselected in a program block with G0 (rapid traverse) or G1 (linear interpolation).**
- **If D0 is programmed when tool radius compensation is active, compensation is not deselected and error message 10750 is output.**
- **If a geometry axis is programmed in the block with the tool radius compensation deselection, then the compensation is deselected even if it is not on the current plane.**
18.5.6 Compensation at outside corners

G450/G451

The G functions G450/G451 can be used to control the response with discontinuous block transitions at outside corners:

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>G450</td>
<td>Discontinuous block transitions with transition circle</td>
</tr>
<tr>
<td>G451</td>
<td>Discontinuous block transitions with intersection of equidistant paths</td>
</tr>
</tbody>
</table>

![Diagram](image)

Figure 18-23 Example of a 90 degree outside corner with G450 and G451

G450 (transition circle)

With the G function G450 active, on outside corners, the center point of the tool travels a circular path along the tool radius. The circular path starts with the normal position (perpendicular to the path tangent) at the end point of the previous path section (program block) and ends in normal position at the start point of the new path section (program block).

Where outside corners are very flat, the response with G450 (transition circle) and G451 (intersection) becomes increasingly similar (see "Very flat outside corners").

If pointed outside corners are desired, the tool must be retracted from the contour (see Section "DISC").
DISC

The G450 transition circle does not produce sharp outside contour corners because the path of the tool center point through the transition circle is controlled so that the cutting edge stops at the outside corner (programmed position). When sharp outside corners are to be machined with G450, the DISC instruction can be used to program an overshoot. Thus, the transition circle becomes a conic and the tool cutting edge retracts from the outside corner.

The range of values of the DISC instruction is 0 to 100, in increments of 1.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISC = 0</td>
<td>Overshoot disabled, transition circle active</td>
</tr>
<tr>
<td>DISC = 100</td>
<td>Overshoot large enough to theoretically produce a response similar to intersection (G451).</td>
</tr>
</tbody>
</table>

The programmable maximum value for DISC can be set via the machine data: MD20220 $MC_CUTCOM_MAX_DISC (max. value for DISC).

Values greater than 50 are generally not advisable with DISC.

![Diagram of DISC overshoot with values 0, 25, and 100](image-url)

Figure 18-24 Example: Overshoot with DISC = 25
18.5 Tool radius compensation 2D (TRC)

**G451 (intersection)**

If G function G451 is active, the position (intersection) resulting from the path lines (straight line, circle or helix only) located at a distance of the tool radius to the programmed contour (center-point path of the tool), is approached. Spines and polynomials are never extended.

**Very pointed outside corners**

Where outside corners are very pointed, G451 can result in excessive idle paths. Therefore, the system switches automatically from G451 (intersection) to G450 (transition circle, with DISC where appropriate) when outside corners are very pointed.

The threshold angle (contour angle) for this automatic switchover (intersection point → transition circle) can be specified in the machine data:

MD20210 $MC_CUTCOM_CORNER_LIMIT (Max. angle for compensation blocks with tool radius compensation).
Very flat outside corners

Where outside corners are very flat, the response with G450 (transition circle) and G451 (intersection) becomes increasingly similar. In this case, it is no longer advisable to insert a transition circle. One reason why it is not permitted to insert a transition circle at these outside corners with 5-axis machining is that this would impose restrictions on speed in contouring mode (G64). Therefore, the system switches automatically from G450 (transition circle, with DISC where appropriate) to G451 (intersection) when outside corners are very flat.

The threshold angle (contour angle) for this automatic switchover (transition circle → intersection point) can be specified in the machine data:

MD20230 $MC_CUTCOM_CURVE_INSERT_LIMIT (Max. angle for intersection calculation with tool radius compensation).
18.5.7 Compensation and inner corners

Intersection

If two consecutive blocks form an inside corner, an attempt is made to find a point at which the two equidistant paths intersect. If an intersection is found, the programmed contour is shortened to the intersection (first block shortened at end, second block shortened at beginning).
No intersection

In certain cases, no intersection is found between two consecutive blocks for inside corners (see figure below).

Predictive contour calculation

If no intersection is found between two consecutive blocks, the control automatically checks the next block and attempts to find an intersection with the equidistant paths of this block (see figure below: intersection S). This automatic check of the next block (predictive contour calculation) is always performed until a number of blocks defined via machine data has been reached.

MD20240 $MC_CUTCOM_MAXNUM_CHECK_BLOCKS (blocks for predictive contour calculation for TRC).

If no intersection is found within the number of blocks defined for the check, program execution is interrupted and an alarm is output.

Figure 18-29 If there is no intersection between N30 and block N40, the intersection between block N30 and block N50 is calculated.
Multiple intersections

It can be the case with inside corners that predictive contour calculation finds multiple intersections of the equidistant paths in several consecutive blocks. In these cases, the last intersection is always used as the valid intersection:

![Diagram of multiple intersections](image)

Figure 18-30 Example: Inside corner with TRC without contour violation (predicting 3 blocks)

For further information, see also Section "Collision detection and bottleneck detection (Page 1430)", "Collision monitoring".

Special features

Where multiple intersections with the next block are found, the intersection nearest the start of the next block applies.

18.5.8 Collision detection and bottleneck detection

Function

Collision detection (bottleneck detection) examines whether the equidistant paths of non-consecutive blocks intersect. If an intersection is found, the response is the same as for inside corners with multiple intersections: The last intersection to be found is valid.

The maximum number of blocks used for the predictive check is set via:

MD20240 $MC_CUTCOM_MAXNUM_CHECK_BLOCKS (blocks for predictive contour calculation for TRC).
Syntax

CDON
CDOF
CDOF2

Meaning

<table>
<thead>
<tr>
<th>CDON:</th>
<th>Collision detection ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDOF:</td>
<td>Collision detection OFF</td>
</tr>
<tr>
<td>CDOF2:</td>
<td>Collision detection OFF, for 3D circumferential milling. CDOF2 has the same significance as CDOF for all other types of machining (e.g. 3D face milling).</td>
</tr>
</tbody>
</table>

Further information

With CDOF, an intersection is sought between two consecutive blocks. If an intersection is found, no further blocks are examined. With outside corners, an intersection is always found between two consecutive blocks.

Predictive examination of more than two adjacent blocks is thus possible with CDON and CDOF.

Omission of block

If an intersection is detected between two blocks which are not consecutive, none of the motions programmed between these blocks on the compensation plane are executed. All other motions and executable instructions (M commands, traversal of positioning axes, etc.) contained in the omitted blocks are executed at the intersection position in the sequence in which they are programmed in the NC program.

Warning 10763: "The path component of the block in the compensation plane will become zero."

If a block has been omitted because of collision detection or bottleneck detection, warning 10763 is displayed. The NC program is not interrupted.

The display of the warning can be suppressed by:

MD11410 $MN_SUPPRES_ALARM_MASK, Bit 1 = 1 (mask supporting special alarm generation)

Bottleneck detection and outside corners

When the intersections of non-consecutive blocks are checked, it is not the programmed original contours that are examined, but the associated calculated equidistant paths. This can result in a "bottleneck" being falsely detected at outside corners. The reason for this is that the calculated tool path does not run equidistant to the programmed original contour when DISC > 0.
18.5.9 Blocks with variable compensation value

Supplementary conditions

A variable compensation value is permissible for all types of interpolation (including circular and spine interpolation).

It is also permitted to change the sign (and, therefore, the compensation side).
**Calculation of intersection**

When the intersections in blocks with variable compensation value are calculated, the intersection of the offset curves (tool paths) is always calculated based on the assumption that the compensation value is constant.

If the block with the variable compensation value is the first of the two blocks to be examined in the direction of travel, then the compensation value at the block end is used for the calculation; the compensation value at the block start is used otherwise.

![Diagram of tool paths and compensation values](image)

**Restrictions**

If during machining on the inside of the circle the compensation radius becomes greater than the programmed circle radius, then the machining is rejected with the following alarm:

Alarm 10758 "Curvature radius with variable compensation value too small"

**Maintain stability of closed contour**

If a radius of two circles is increased slightly, a third block may be necessary in order to maintain the stability of the closed contour. This is the case if two adjacent blocks, which represent two possible intersection points for a closed contour, are skipped due to the compensation.

A stable closed contour can be achieved by choosing the first intersection point instead of the second.

SD42496 $SC\_CUTCOM\_CLSD\_CONT \neq 0$ (response of TRC for closed contour).

In that case, the second intersection point is always reached, even if the block is skipped. A third block is then not required.
18.5.10 Keep tool radius compensation constant

Description

The "Keep tool radius compensation constant" function is used to suppress tool radius compensation for a number of blocks, whereby a difference between the programmed and the actual tool center path traveled set up by tool radius compensation in the previous blocks is retained as the compensation.

It can be an advantage to use this method when several traversing blocks are required during line milling in the reversal points, but the contours produced by the tool radius compensation (follow strategies) are not wanted.

Activation

The "Keep tool radius compensation constant" function is activated with the G code 

CUTCONON (CUTter compensation CONstant ON) and deactivated with the G code 

CUTCONOF (CUTter compensation CONstant OFF).

CUTCONON and CUTCONOF form a modal G-code group.

The initial setting is CUTCONOF.

The function can be used independently of the type of tool radius compensation (2\(\frac{1}{2}\)D, 3D face milling, 3D circumferential milling).

Normal case

Tool radius compensation is normally active before the compensation suppression and is still active when the compensation suppression is deactivated again.

In the last traversing block before CUTCONON, the offset point in the block end point is approached. All following blocks, in which compensation suppression is active, are traversed without compensation. However, they are offset by the vector from the end point of the last offset block to its offset point. These blocks can have any type of interpolation (linear, circular, polynomial).

The deactivation block of the compensation suppression, i.e. the block that contains 

CUTCONOF, is compensated normally. It starts in the offset point of the start point. One linear block is inserted between the end point of the previous block, i.e. the last programmed traversing block with active CUTCONON, and this point.

Circular blocks, for which the circle plane is perpendicular to the compensation plane (vertical circles), are treated as though they had CUTCONON programmed. This implicit activation of compensation suppression is automatically canceled in the first traversing block that contains a traversing motion in the compensation plane and is not such a circle. Vertical circle in this sense can only occur during circumferential milling.

Example:

```
N10 ; Definition of tool d1
N20 $TC_DP1[1,1] = 110 ; Type
N30 $TC_DP6[1,1] = ; Radius
N40
```
### Special cases

- **If tool radius compensation is not active** (G40), `CUTCONON` has no effect. No alarm is produced. The G code remains active, however. This is significant if tool radius compensation is to be activated in a later block with G41 or G42.

- **It is permissible to change the G code in the 7th G-code group** (tool radius compensation; G40 / G41 / G42) with `CUTCONON` active. A change to G40 is active immediately. The offset used for traversing the previous blocks is traveled.

- **If `CUTCONON` or `CUTCONOF` is programmed in a block without traversing in the active compensation plane**, activation is delayed until the next block that has such a traversing motion.

---

**Figure 18-34**  Sample program for contour suppression
If \texttt{CUTCONON} is programmed with active tool radius compensation and not canceled before the end of the program, the traversing blocks are traversed with the last valid compensation. The same applies for reprogramming of \texttt{G41} or \texttt{G42} in the last traversing block of a program.

If tool radius compensation is activated with \texttt{G41} or \texttt{G42} and \texttt{CUTCONON} is also already active, activation of compensation is delayed until the next traversing block with \texttt{CUTCONOF}.

When reapproaching the contour with \texttt{CUTCONOF}, the 17th G-code group (approach and retraction behavior with tool compensation; \texttt{NORM / KONT}) is evaluated, i.e. a bypass circle is inserted if necessary for \texttt{KONT}. A bypass circle is inserted under the same conditions as for activation of tool radius compensation with \texttt{G41} or \texttt{G42}.

The number of blocks with suppressed tool radius compensation is restricted: \texttt{MD20252 $MC_CUTCOM_MAXNUM_SUPPR_BLOCKS} (Maximum number of blocks with compensation suppression). If it is exceeded, machining is aborted and an error message issued. The restriction is necessary because the internal block processing in the last block before \texttt{CUTCONON} must be resumed when repositioning.

The response after reprogramming \texttt{G41} or \texttt{G42} when tool radius compensation is already active is similar to compensation suppression. The following deviations apply:

- Only linear blocks are permissible
- A single traversing block that contains \texttt{G41} or \texttt{G42} is modified so that it ends at the offset point of the start point in the following block. Thus it is not necessary to insert a dummy block. The same applies to the last block in a sequence of traversing blocks where each contains \texttt{G41} or \texttt{G42}.
- The contour is always reapproached with \texttt{NORM}, independent of the G code of the 17th group (approach and retraction behavior with tool compensation; \texttt{NORM / KONT}).

If \texttt{G41 / G42} is programmed several times in consecutive traversing blocks, all blocks are machined as for \texttt{CUTCONON}, except for the last one.

The type of contour suppression is evaluated only in the first traversing block of a sequence of consecutive traversing blocks. If both \texttt{CUTCONON} and \texttt{G41} or \texttt{G42} are programmed in the first block, the response to deactivating contour suppression is determined by \texttt{CUTCONON}. Changing from \texttt{G41} to \texttt{G42} or vice versa makes sense in this case as a means of changing the compensation side (left or right of the contour) when restarting. A change of compensation side (\texttt{G41/G42}) can also be programmed in a later block, even if contour suppression is active.

Collision detection and bottleneck detection is deactivated for all blocks with active contour suppression.
18.5.11 Alarm behavior

Alarm during preprocessing

If a tool radius compensation alarm is output during preprocessing, main-run machining stops at the next block end reached, i.e. usually at the end of the block currently being interpolated (if Look Ahead is active, once the axes have come to a stop).

Alarms for preprocessing stop and active tool radius compensation

Tool radius compensation generally requires at least one of the following traversing blocks (even more for bottlenecks) to determine the end point of a block. Since the preprocessing stop of such a block is not available, traversing continues to the offset point in the last block. Correspondingly, the offset point in the start point is approached in the first block after a preprocessing stop.

The contour obtained may deviate considerably from the one that would result without preprocessing stop. Contour violations in particular are possible. Therefore the following setting data was introduced:

\[ \text{SD42480 \$MC\_STOP\_CUTCOM\_STOPRE (alarm response for TRC and preprocessing stop).} \]

The response of the tool radius compensation remains unchanged compared to the previous status, and/or an alarm is output for preprocessing stop during active tool radius compensation and the program is halted, depending on the value.

The user can acknowledge this alarm and continue the NC program with NC start or abort it with \text{RESET}.

18.5.12 Intersection procedure for polynomials

Function

If two curves with active tool radius compensation form an outside corner, depending on the G code of the 18th group (corner behavior with tool compensation; \text{G450} / \text{G451}) and regardless of the type of curves involved (straight lines, circles, polynomials):

- Either a conic is inserted to bypass the corner

  Or

- The curves involved are extrapolated to form an intersection.

If no intersection is found with \text{G451} activated, or if the angle formed by the two curves is too steep, switchover to insert mode is automatic.

The intersection procedure for polynomials is released with the machine data:
18.5 Tool radius compensation 2D (TRC)

MD20256 $MC_CUTCOM_INTERS_POLY_ENABLE (Intersection process possible for polynomials)

**Note**
If this machine data is set to inactive, a block (can be very short) is always inserted (even if transitions are almost tangential). These short blocks always produce unwanted drops in speed during G64 operation.

### 18.5.13 G461/G462 Approach/retract strategy expansion

**Function**
In certain special geometrical situations, extended approach and retraction strategies, compared with the previous implementation, are required in order to activate or deactivate tool radius compensation (see figure below).

**Note**
The following example describes only the situation for deactivation of tool radius compensation. The response for approach is virtually identical.

**Example**
```
G42 D1 T1 ; Tool radius 20 mm
...
G1 X110 Y0
N10 X0
N20 Y10
N30 G40 X50 Y50
```
The last block with active tool radius compensation (N20) is so short that an intersection no longer exists between the offset curve and the preceding block (or a previous block) for the current tool radius. An intersection between the offset curves of the following and preceding blocks is therefore sought, i.e., between N10 and N30 in this example. The curve used for the retraction block is not a real offset curve, but a straight line from the offset point at the end of block N20 to the programmed end point of N30. The intersection is approached if one is found. The colored area in the figure is not machined, although the tool used would be capable of this.

G460

With **G460**, the approach/retraction strategy is the same as before.
G461

If no intersection is possible between the last TRC block and a preceding block, the offset curve of this block is extended with a circle whose center point lies at the end point of the uncorrected block and whose radius is equal to the tool radius.

Figure 18-36 Retraction behavior with G461

The control attempts to cut this circle with one of the preceding blocks. If CDOP is active, the search is terminated when an intersection is found, i.e., the system does not check for more intersections with even earlier blocks.

If CDON is active, the search for more intersections continues after the first intersection is found.

An intersection point, which is found in this way, is the new end point of a preceding block and the start point of the deactivation block. The inserted circle is used exclusively to calculate the intersection and does not produce a traversing movement.

**Note**

If no intersection is found, the following alarm is output:

Alarm "10751 Collision danger"
G462

If no intersection is possible between the last TRC block and a preceding block, a straight line is inserted, on retraction with G462 (initial setting), at the end point of the last block with tool radius compensation (the block is extended by its end tangent).

![Diagram of retraction behavior with G462](image)

Figure 18-37 Retraction behavior with G462

The search for the intersection is then identical to the procedure for G461.

With G462, the corner generated by N10 and N20 in the sample program is not machined to the full extent actually possible with the tool used. However, this behavior may be necessary if the part contour (as distinct from the programmed contour), to the left of N20 in the example, is not permitted to be violated even with y values greater than 10 mm.

If KONT is active (travel round contour at start or end point), behavior will differ according to whether the end point is in front of or behind the contour.

**End point in front of contour**

If the end point is located in front of the contour, the retraction behavior is the same as for NORM. This feature does not change, even if the last contour block with G451 is extended with a straight line or a circle. Additional circumnavigation strategies to avoid a contour violation in the vicinity of the contour end point are therefore not required.

**End point behind contour**

If the end point is behind the contour, a circle or straight line is always inserted depending on G450/G451. In this case, G460-G462 has no effect.

If, in this situation, the last traversing block has no intersection with a preceding block, an intersection with the inserted contour element or with the linear section from the end point of the circumnavigation circle to the programmed end point can result.

If the inserted contour element is a circle (G450), and it intersects with the preceding block, this is the same as the intersection, which would be produced with NORM and G461. In general, however, a remaining section of the circle still has to be traversed. An intersection calculation is no longer required for the linear section of the retraction block.
In the second case (if no intersection is found between the inserted contour element and the preceding blocks), the intersection between the retraction straight line and a preceding block is approached.

Therefore, when G461 or G462 is active, behavior deviating from G460 can only arise if NORM is active or if behavior with KONT is identical to NORM due to the geometrical conditions.

**Note**

The approach behavior is symmetrical to the retraction behavior.

The approach/retraction behavior is determined by the state of the G command in the approach/retraction block. The approach behavior can therefore be set independently of the retraction behavior.

**Example:**

Program for using G461 during approach:

```
N10 $TC_DP1[1,1]=120 ; Tool type mill
N20 $TC_DP6[1,1]=10 ; Radius
N30 X0 Y0 F10000 T1 D1
N40 Y20
N50 G42 X50 Y5 G461
N60 Y0 F600
N70 X30
N80 X20 Y-5
N90 X0 Y0 G40
N100 M30
```

**18.6 Toolholder with orientation capability**

**18.6.1 General**

**Introduction**

The orientation of the tool can vary (e.g. due to retooling) for one single class of machine tools. When the machine is operating, the orientation that has been set is permanent, however, and cannot be changed during traversing. For this reason, kinematic orientation transformation (3-, 4- or 5-axis transformations, TRAORI) is neither necessary nor does it make sense for such machines. However, it is necessary to take account of the changes in the tool length components caused by changing the orientation, without having to trouble the user with mathematics involved. The control performs these calculations.
Availability

For SINUMERIK 828D, the "toolholder with orientation capability" function is only available for the milling versions.

Required data

The following requirements must be met if the control is to take tool compensations into account for toolholders with orientation capability:

- Tool data (geometry, wear, etc.)
- Toolholder data (data for the geometry of the toolholder with orientation capability)

Toolholder selection

A toolholder defined in the control must be specified for the "Toolholder with orientation capability" function. The NC program command below is used for this purpose:

\[ \text{TCA} \text{arr} = m \]

\( m \): Number of the toolholder

The toolholder has an associated toolholder data block that describes its geometry.

Activating the toolholder and its block has an immediate effect, i.e. from the next traversing block onwards.

Assignment tool/toolholder

The tool that was active previously is assigned to the new toolholder.

From the point of view of the control, toolholder number \( m \) and tool number \( T \) can be combined freely. In the real application, however, certain combinations can be ruled out for machining or mechanical reasons. The control does not check whether the combinations make sense.

Description of the kinematics of the toolholder

The kinematics of the toolholder with orientation capability is described with a total of 33 parameter sets.

The data of the data block can be edited by the user.
Toolholder with orientation capability

Example: Cardan toolholder with two axes for the tool orientation

![Cardan toolholder with two axes](image)

Figure 18-38 Cardan toolholder with two axes

Processing toolholder data blocks

Two options are available:

- Explicit entry in the toolholder data block from the part program
- Automatic acceptance of certain values (angles) from a frame

A requirement for this is that $TCOFR$ (Tool Carrier Orientation Frame) is also specified when the toolholder is selected.

The tool orientation used to calculate the tool length is determined again from the frame active at this time when a toolholder is changed.

Orientation in Z direction

The G function $TOFRAME$ defines a frame such that the Z direction in this frame is the same as the current tool orientation.

If no toolholder or a toolholder without change in orientation is active, then the Z direction is in the new frame:

- The same as the old Z direction with $G17$.
- The same as the old Y direction with $G18$.
- The same as the old X direction with $G19$. 
TCOABS for active frame

The absolute toolholder orientation is set using:

TCOABS (Tool Carrier Orientation ABSolute)

The orientation taken into account for the tool length compensation is independent of the orientation of the active frame.

Only one of the commands TCOABS or TCOFR can be valid.

Frame change

The user can change the frame following selection of the tool. This does not have any effect on the tool length compensation components.

Angles in the toolholder data:

The programmed angles of rotation stored in the toolholder data is not affected by the angle of rotation defined by the frames. When changing from TCOFR to TCOABS, the original (programmed) angles of rotation in the toolholder data is reactivated.

Tool compensation types

TRC takes account of the current tool orientation when CUT2D or CUT3DFS is active.

All other tool compensation types

These are all the compensation types of G-code group 22, with the exception of CUT3DC and CUT3DF. The response remains the same with respect to the plane used for compensation. This is determined independent of the tool orientation from the active frame.

For CUT2DF and CUT3DF, the compensation plane used for TRC is determined from the frame independently of the current tool orientation. The active plane (G17/G18/G19) is considered.

CUT3DC and CUT3DF

3D tool compensation for circumferential milling

3D tool compensations for face milling with active 5-axis transformation are not affected by the "Toolholder with orientation capability" function.

The orientation information is determined by the active kinematic 5-axis transformation.

Limited toolholder orientation

An alarm is output if an orientation that cannot be reached with the defined toolholder kinematic is specified by the frame.

The following kinematics cannot achieve any orientation:

- If the two rotary axes which are necessary to define the kinematics are not perpendicular to each other and if the tool axis which defines the tool direction is not perpendicular to the second rotary axis or
- Less than two axes have been defined
Non-rotary toolholders

The tool orientation used internally is dependent only on the basic orientation of the tool and the active plane (G17 - G19).

Ambiguities

With two axes, a particular tool orientation defined by the frame can generally be set with two different rotary angle pairs. Of these two, the control selects the setting with which the rotary angle is as close as possible to the programmed rotary angle.

Storing angles in the toolholder data

In virtually any case where ambiguities may arise, it is necessary to store the approximate angle expected from the frame in the toolholder data.

Parameter sets

A complete parameter set for a toolholder with orientation capability consists of 33 values.

The following system variables are available:

- $TC_CARR1 to $TC_CARR33
- In addition, $TC_CARR34 to $TC_CARR65 are freely available for the user and for fine offsets.

The significance of the individual parameters is distinguished as follows:

**Machine kinematics:**

$TC_CARR1 to $TC_CARR20 and $TC_CARR23

$TC_CARR18 to $TC_CARR20 define a further vector \( \mathbf{l}_4 \) which is needed to describe the machine with extended kinematics (both tool and workpiece can be rotated).

$TC_CARR21 and $TC_CARR22 contain the channel-axis names of the rotary axes, the positions of which can be used to determine the orientation of the toolholder with orientation capability, if necessary.

**Kinematic type:**

$TC_CARR23 using letter T, P or M

The following three options are available for the kinematic type, for which both upper and lower case text are permissible:

<table>
<thead>
<tr>
<th>Character (T, P, M)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T:</td>
<td>Only the (Tool) can be rotated (basic value).</td>
</tr>
<tr>
<td>P:</td>
<td>Only the workpiece (Part) can be rotated.</td>
</tr>
<tr>
<td>M:</td>
<td>Both tool and workpiece can be rotated (Mixed mode).</td>
</tr>
</tbody>
</table>

Any character other than the three mentioned here will result in an alarm if it is tried to activate the toolholder with orientation capability:

Alarm "14153 Channel %1 block %2 unknown toolholder type: %3"

**Rotary axis parameters:**
$TC_CARR24 to $TC_CARR33

The system variables in $TC_CARR24 to $TC_CARR33 can be used to define offsets, angle compensations, Hirth tooth system and axis limits.

**Note**

The system variables are available with and without active tool management.

**Components and presetting of the chain/data block**

The values $TC_CARR1 to $TC_CARR20 and $TC_CARR24 to $TC_CARR33 in the toolholder data block are of NC language format type REAL.

The values $TC_CARR21 and $TC_CARR22 for the axis name of the first rotary axis (v₁) and the second rotary axis (v₂) are of NC language format type AXIS. They are all preset to zero.

The value $TC_CARR23 is initialized with the uppercase letter "T" (only tool can be rotated).

$TC_CARRn[m]

$TC_CARR[0]= 0 has a special significance

**System variables for toolholders with orientation capability**

$TC_CARRn[m]

<table>
<thead>
<tr>
<th>Description</th>
<th>NCK variable</th>
<th>Language format</th>
<th>Default setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>x component of offset vector l₁</td>
<td>$TC_CARR1</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>y component of offset vector l₁</td>
<td>$TC_CARR2</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>z component of offset vector l₁</td>
<td>$TC_CARR3</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>x component of offset vector l₂</td>
<td>$TC_CARR4</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>y component of offset vector l₂</td>
<td>$TC_CARR5</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>z component of offset vector l₂</td>
<td>$TC_CARR6</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>x component of rotary axis v₁</td>
<td>$TC_CARR7</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>y component of rotary axis v₁</td>
<td>$TC_CARR8</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>z component of rotary axis v₁</td>
<td>$TC_CARR9</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>x component of rotary axis v₂</td>
<td>$TC_CARR10</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>y component of rotary axis v₂</td>
<td>$TC_CARR11</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>z component of rotary axis v₂</td>
<td>$TC_CARR12</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Angle of rotation α₁ (in degrees)</td>
<td>$TC_CARR13</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Angle of rotation α₂ (in degrees)</td>
<td>$TC_CARR14</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Description</td>
<td>NCK variable</td>
<td>Language format</td>
<td>Default setting</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>x component of offset vector $l_1$</td>
<td>$TC_CARR15$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>y component of offset vector $l_1$</td>
<td>$TC_CARR16$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>z component of offset vector $l_1$</td>
<td>$TC_CARR17$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>x component of offset vector $l_2$</td>
<td>$TC_CARR18$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>y component of offset vector $l_2$</td>
<td>$TC_CARR19$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>z component of offset vector $l_2$</td>
<td>$TC_CARR20$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Axis name of the rotary axis $v_1$</td>
<td>$TC_CARR21$</td>
<td>AXIS</td>
<td>0</td>
</tr>
<tr>
<td>Axis name of the rotary axis $v_2$</td>
<td>$TC_CARR22$</td>
<td>AXIS</td>
<td>0</td>
</tr>
<tr>
<td>Kinematic type</td>
<td>$TC_CARR23$</td>
<td>CHAR</td>
<td>T</td>
</tr>
<tr>
<td>Offset of rotary axis $v_1$</td>
<td>$TC_CARR24$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Offset of rotary axis $v_2$</td>
<td>$TC_CARR25$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Angle offset of rotary axis $v_1$ (Hirth tooth)</td>
<td>$TC_CARR26$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Angle offset of rotary axis $v_2$ (Hirth tooth)</td>
<td>$TC_CARR27$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Angle increment of rotary axis $v_1$ (Hirth tooth)</td>
<td>$TC_CARR28$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Angle increment of rotary axis $v_2$ (Hirth tooth)</td>
<td>$TC_CARR29$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Minimum position of rotary axis $v_1$ (SW limit)</td>
<td>$TC_CARR30$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Minimum position of rotary axis $v_2$ (SW limit)</td>
<td>$TC_CARR31$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Maximum position of rotary axis $v_1$ (SW limit)</td>
<td>$TC_CARR32$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Maximum position of rotary axis $v_2$ (SW limit)</td>
<td>$TC_CARR33$</td>
<td>REAL</td>
<td>0</td>
</tr>
</tbody>
</table>

**System variables for the user and for fine offsets**

- $TC\_CARR34$ to $TC\_CARR40$
  Contain parameters, which are freely available to the user.
- $TC\_CARR41$ to $TC\_CARR65$
  Contain fine offset parameters that can be added to the values in the basic parameters. The fine offset value assigned to a basic parameter is obtained when the value 40 is added to the parameter number.
- $TC\_CARR47$ to $TC\_CARR54$ and $TC\_CARR61$ to $TC\_CARR63$
  Not defined and produce an alarm if read or write access is attempted.

<table>
<thead>
<tr>
<th>Description</th>
<th>NCK variable</th>
<th>Language format</th>
<th>Default setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toolholder name</td>
<td>$TC_CARR34$</td>
<td>String[32]</td>
<td>***</td>
</tr>
<tr>
<td>Axis name 1</td>
<td>$TC_CARR35$</td>
<td>String[32]</td>
<td>***</td>
</tr>
<tr>
<td>Axis name 2</td>
<td>$TC_CARR36$</td>
<td>String[32]</td>
<td>***</td>
</tr>
<tr>
<td>Identifier</td>
<td>$TC_CARR37$</td>
<td>INT</td>
<td>0</td>
</tr>
<tr>
<td>Position component X</td>
<td>$TC_CARR38$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Position component Y</td>
<td>$TC_CARR39$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Position component Z</td>
<td>$TC_CARR40$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>x comp. fine offset of offset vector $l_1$</td>
<td>$TC_CARR41$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>y comp. fine offset of offset vector $l_1$</td>
<td>$TC_CARR42$</td>
<td>REAL</td>
<td>0</td>
</tr>
</tbody>
</table>
### 18.6.2 Kinematic interaction and machine design

#### Representation of the kinematic chain

The concept of the kinematic chain is used to describe the kinematic interaction between a reference point and the tool tip.

The chain specifies all the data required for the toolholder data block in a schematic. To describe the concrete case with a particular kinematic, the relevant components of the chain must be assigned real vectors, lengths and angles. The chain represents the maximum constellation. In simpler applications, individual components can be zero (e.g. kinematics with one or no rotary axis).

The machine does not have to have axes that rotate the tool and/or workpiece table. The function can be used even if the orientations are set manually by handwheels or reconfiguration.

The machine design is described by the following parameters:

- Two rotary axes ($v_1$ and $v_2$), each with one angle of rotation ($\alpha_1$ or $\alpha_2$), which counts positively for clockwise rotation facing the direction of the rotation vector.
- Up to four offset vectors ($l_1$ to $l_4$) for relevant machine dimensions (axis distances, distances to machine or tool reference points).

#### Zero vectors

Vectors $v_1$ and $v_2$ can be zero. The associated angle of rotation (explicitly programmed or calculated from the active frame) must then also be zero, since the direction of the rotating axis is not defined. If this condition is not satisfied, an alarm is produced when the toolholder is activated.

<table>
<thead>
<tr>
<th>Description</th>
<th>NCK variable</th>
<th>Language format</th>
<th>Default setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>z comp. fine offset of offset vector l z</td>
<td>$TC_CARR43$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>x comp. fine offset of offset vector l x</td>
<td>$TC_CARR44$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>y comp. fine offset of offset vector l y</td>
<td>$TC_CARR45$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>z comp. fine offset of offset vector l z</td>
<td>$TC_CARR46$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>x comp. fine offset of offset vector l x</td>
<td>$TC_CARR55$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>y comp. fine offset of offset vector l y</td>
<td>$TC_CARR56$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>z comp. fine offset of offset vector l z</td>
<td>$TC_CARR57$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>x comp. fine offset of offset vector l x</td>
<td>$TC_CARR58$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>y comp. fine offset of offset vector l y</td>
<td>$TC_CARR59$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>z comp. fine offset of offset vector l z</td>
<td>$TC_CARR60$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Offset of fine offset of rotary axis $v_1$</td>
<td>$TC_CARR64$</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>Offset of fine offset of rotary axis $v_2$</td>
<td>$TC_CARR65$</td>
<td>REAL</td>
<td>0</td>
</tr>
</tbody>
</table>

Remarks:

*) The system variables $TC\_CARR35$ to $TC\_CARR40$ are used in the measuring cycles as well as ShopMill and ShopTurn.
Less than two rotating axes

The option not to define a rotating axis makes sense when the toolholder to be described can only rotate the tool in one plane. A sensible minimum data block may, therefore, contain only one single entry not equal to 0 in the toolholder data; namely, a value in one of the components of \( v_1 \) or \( v_2 \) for describing a rotating axis parallel to the axis where the angle of rotation \( \alpha_1 \) or \( \alpha_2 \) is determined from one frame.

Further special cases

Vectors \( v_1 \) and \( v_2 \) can be colinear. However, the degree of freedom for orientation is lost, i.e. this type of kinematic is the same as one where only one rotary axis is defined. All possible orientations lie on one cone sheath. The conical sheath deforms to a straight line if tool orientation \( t \) and \( v_1 \) or \( v_2 \) become colinear. Change of orientation is, therefore, not possible in this special case. The cone sheath deforms to a circular surface (i.e. all orientations are possible in one plane), if tool orientation \( t \) and \( v_1 \) or \( v_2 \) are perpendicular to each other.

It is permissible for the two vectors \( v_1 \) and \( v_2 \) to be zero. A change in orientation is then no longer possible. In this special case, any lengths \( l_1 \) and \( l_2 \), which are not equal to zero, act as additional tool length compensations, in which the components in the individual axes are not affected by changing the plane \( (G17 - G19) \).

Kinematics data expansions

- Possibility of direct access to existing machine axes in order to define the toolholder setting via the rotary axis positions.
- Extension of the kinematics with rotary workpiece and on kinematics with rotary tool and rotary workpiece.
- Possibility to permit only discrete values in a grid for the rotary axis positions (Hirth tooth system).

The extensions are compatible with earlier software versions and encompass the kinematic data blocks from $TC_CARR18 to $TC_CARR23.

Machine with rotary tool

On machines with rotary tool there is no change in the definition of the kinematics compared to older software versions. The newly introduced vector \( l_4 \), in particular, has no significance. Should the contents of \( l_4 \) not be zero, this is ignored.

The term "Toolholder with orientation capability" is actually no longer really appropriate for the new kinematic types, with which the table can also be rotated, either alone or additionally to the tool. However, it has been kept for reasons of compatibility.

The kinematic chains used to describe the machine with rotary tool (general case) are shown in the figure below:
Vectors, which describe offsets in the rotary head, are positive in the direction from the tool tip to the reference point of the toolholder.

The following kinematic type is defined for machines with a rotary tool:

$\text{STC\_CARR23 using letter T}$

**Machine with rotary workpiece**

On machines with rotary workpiece, the vector $l_1$ has no significance. If it contains a value other than zero, this is ignored.

The kinematic chain for machines with rotary workpiece is shown in the figure below.
Vectors, which describe offsets in the rotary table, are positive in the direction from the machine reference point to the table.

The following kinematic type is defined for machines with a rotary workpiece:

$\text{TC\_CARR23 using letter P}$

**Note**

On machines with rotary workpiece it is generally useful if the selected machine reference point and the reference point of the table are identical. Selecting the reference points in this way has the advantage that the position of the workpiece zero in the initial state (i.e. with rotary axes not turned) does not change when the rotary table is activated. The (open) kinematic chain (see figure) is then closed.

In this special case, therefore, the following formula applies: $l_2 = -(l_3 + l_4)$

---

**Machines with extended kinematics**

On machines with extended kinematics (both tool and workpiece are rotary), it is only possible to turn each of the components with one axis.

The kinematic of the rotary tool is described with the first rotary axis ($v_1$) and the two vectors $l_1$ and $l_2$, that of the rotary table with the second rotary axis ($v_2$) and the two vectors $l_3$ and $l_4$. The two kinematic chain components for machines with rotary tool and rotary workpiece are shown in the figure below.
The following kinematic type is defined for machines with a rotary tool and rotary workpiece:

$\text{TC}_C\text{ARR}23$ using letter $M$ (extended kinematics)

**Note**

On machines with extended kinematics it is generally useful, as with machines where only the table can be rotated, for the machine reference point and the reference point of the table to be identical. The (open) chain component to describe the table (see figure) is then closed.

In this special case, the following formula applies: $l_3 = -l_4$

**Rotary tool types T and M**

For machine kinematics with a rotary tool (types T and M), the toolholder component with orientation capability, which describes the tool or head component (as opposed to the table component), acts, in conjunction with the active tool, as a new overall tool.

**Fine offset**

The offset vectors $l_1$ to $l_4$ and the offsets of the rotary axes $v_1$ and $v_2$ can be represented as the sum of a basic value and a fine offset. The fine offset parameters assigned to the basic values are achieved by adding a value of 40 to the index of the basic value.

**Example:**
The parameter $TC_CARR5$ is assigned to the fine offset $TC_CARR45$.

**Note**
For the significance of the system variables $TC_CARR41$ to $TC_CARR65$ available for the fine offset see:

**References:**
Programming Manual, Job Planning; Tool Offsets:

**Activation**
The following setting adds the fine offset values to the basic values:

SD42974 $SC_TOCARR_FINE_CORRECTION = 1$ (fine offset TCARR on/off)

**Supplementary conditions**
The amount is limited to the permissible fine offset.
The maximum permissible value is defined:

<table>
<thead>
<tr>
<th>For:</th>
<th>With machine data:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The components of vectors $l_1$ to $l_4$:</td>
<td>MD20188 $MC_TOCARR_FINE_LIM_LIN</td>
</tr>
<tr>
<td>The offsets of the two rotary axes $v_1$ and $v_2$:</td>
<td>MD20190 $MC_TOCARR_FINE_LIM_ROT</td>
</tr>
</tbody>
</table>

An illegal fine offset value is only detected when:
- A toolholder with orientation capability, which contains such a value, is activated and
- at the same time the following setting data is set:
  SD42974 $SC_TOCARR_FINE_CORRECTION$

**Description of a rotation**
A data block for describing a rotation comprises one vector $v_1/v_2$ to describe the direction of rotation of the rotary axis in its initial state and an angle $\alpha_1/\alpha_2$. The angle of rotation is counted positively for clockwise rotation facing the direction of the rotation vector.

The two toolholder angles $\alpha_1$ and $\alpha_2$ are determined using a frame, independent of the active plane currently selected ($G17 - G19$).

The tool orientation in the initial state (both angles $\alpha_1$ and $\alpha_2$ are zero) is (as in the default case):
- $G17$: Parallel to $Z$.
- $G18$: Parallel to $Y$.
- $G19$: Parallel to $Z$

**Assigning data to the toolholder**

**Example of a machine with rotary toolholder**
The following settings are obtained at the mill head shown for a machine with toolholder with orientation capability of kinematic type T:

| Component of the offset vector $l_1$ | (-200, 0, 0) |
| Component of the offset vector $l_2$ | (0, 0, 0) |
| Component of the offset vector $l_3$ | (-100, 0, 0) |
| Component of rotary axis $v_1$ | (1, 0, 0) |
| Component of rotary axis $v_2$ | (-1, 0, 1) |
| Tool base dimension of tool reference point | (0, 0, 250) |

**Note**

The tool reference point for the tool base dimension is defined by the reference point at the machine.

For further information about the reference points in the working area, see Section "K2: Axis Types, Coordinate Systems, Frames (Page 661)".

Suitable assumptions were made for the following values in the data block:

- The two rotary axes intersect at one point. All components of $l_2$ are therefore zero.

- The first rotary axis lies in the x/z plane, the second rotary axis is parallel to the x axis. These conditions define the directions of $v_1$ and $v_2$ (the lengths are irrelevant, provided that they are not equal to zero).

- The reference point of the toolholder lies 200 mm in the negative x direction viewed from the intersection of the two rotary axes. This condition defines $l_1$. 

Figure 18-42 Assignment of the toolholder data
Specify associated data block values

The following associated data block values are specified for the toolholder shown on a machine with rotary toolholder:

<table>
<thead>
<tr>
<th>Description</th>
<th>NCK variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x component of offset vector l₁</td>
<td>$TC_CARR1</td>
<td>-200</td>
</tr>
<tr>
<td>y component of offset vector l₁</td>
<td>$TC_CARR2</td>
<td>0</td>
</tr>
<tr>
<td>z component of offset vector l₁</td>
<td>$TC_CARR3</td>
<td>0</td>
</tr>
<tr>
<td>x component of offset vector l₂</td>
<td>$TC_CARR4</td>
<td>0</td>
</tr>
<tr>
<td>y component of offset vector l₂</td>
<td>$TC_CARR5</td>
<td>0</td>
</tr>
<tr>
<td>z component of offset vector l₂</td>
<td>$TC_CARR6</td>
<td>0</td>
</tr>
<tr>
<td>x component of rotary axis v₁</td>
<td>$TC_CARR7</td>
<td>1</td>
</tr>
<tr>
<td>y component of rotary axis v₁</td>
<td>$TC_CARR8</td>
<td>0</td>
</tr>
<tr>
<td>z component of rotary axis v₁</td>
<td>$TC_CARR9</td>
<td>0</td>
</tr>
<tr>
<td>x component of rotary axis v₂</td>
<td>$TC_CARR10</td>
<td>-1</td>
</tr>
<tr>
<td>y component of rotary axis v₂</td>
<td>$TC_CARR11</td>
<td>0</td>
</tr>
<tr>
<td>z component of rotary axis v₂</td>
<td>$TC_CARR12</td>
<td>1</td>
</tr>
<tr>
<td>Angle of rotation α₁ (in degrees)</td>
<td>$TC_CARR13</td>
<td>0</td>
</tr>
<tr>
<td>Angle of rotation α₂ (in degrees)</td>
<td>$TC_CARR14</td>
<td>0</td>
</tr>
<tr>
<td>x component of offset vector l₃</td>
<td>$TC_CARR15</td>
<td>-100</td>
</tr>
<tr>
<td>y component of offset vector l₃</td>
<td>$TC_CARR16</td>
<td>0</td>
</tr>
<tr>
<td>z component of offset vector l₃</td>
<td>$TC_CARR17</td>
<td>0</td>
</tr>
</tbody>
</table>

Explanations

The toolholder kinematic chosen in the example is such that the two rotary axes form an angle of 45 degrees, which means that the orientation cannot take just any value. In concrete terms, this example does not permit the display of orientations with negative X components.

x component of the tool base dimension: 0
y component of the tool base dimension: 0
z component of the tool base dimension: 250

Note

The required data cannot be determined unequivocally from the geometry of the toolholder, i.e. the user is free to a certain extent to decide the data to be stored. Thus, for the example, it is possible to specify only one z component for the tool base dimension up to the second axis. In this case, l₂ would no longer be zero, but would contain the components of the distance between this point on the second axis and a further point on the first axis. The point on the first axis can also be selected freely. Depending on the point selected, l₁ must be selected such that the reference point (which can also be selected freely) is reached.

In general: vector components that are not changed by rotation of an axis can be distributed over any vectors "before" and "after" rotation.
18.6.3 Inclined surface machining with 3 + 2 axes

Description of function

Inclined machining with 3 + 2 axes describes an extension of the concept of toolholders with orientation capability and applies this concept to machines with a rotary table, on which the orientation of tool and table can be changed simultaneously.

The "Inclined machining with 3 + 2 axes" function is used to machine surfaces with any rotation with reference to the main planes X/Y (G17), Z/X (G18) and Y/Z (G19).

It is possible to produce any orientation of the tool relative to the workpiece by rotating either the tool, the workpiece or both the tool and the workpiece.

The software automatically calculates the necessary compensating movements resulting from the tool lengths, lever arms and the angle of the rotary axis. It is always assumed that the required orientation is set first and not modified during a machining process such as pocket milling on an inclined plane.

Furthermore, the following 3 functions are described, which are required for oblique machining:

- **Position programming** in the direction of the tool orientation independent of an active frame
- **Definition of a frame rotation** by specifying the solid angle
- **Definition of the component of rotation in tool direction** in the programmed frame while maintaining the remaining frame components

Demarcation to 5-axis transformation

If the required functionality specifies that the TCP (Tool Center Point) does not vary in the event of reorientation with reference to the workpiece, even during interpolation, the 5-axis software is required.

For more explanations on 5-axis transformations, see:

**References:**
Function Manual, Special Functions; 3- to 5-Axis Transformation (F2)

Specification of the toolholder with orientation capability

The toolholder with orientation capability is represented by a general 5axis kinematic sequence described by a data block in the tool compensation memory with a total of 33 REAL values. For toolholders that have two rotary axes for setting the orientation (e.g. a millhead), 31 of these values are constant.

In the current SW version, a data block in the tool compensation memory is described with a total of 47 REAL values. For toolholders that have two rotary axes for setting the orientation, 45 of these values are constant.

The remaining two values are variable and are used to specify the orientation. The constant values describe offsets and directions and setting options for the rotary axes; the variable values describe the angles of the rotary axes.
18.6.4 Machine with rotary work table

System variables

To date, the angles stored in $TC_CARR13 and $TC_CARR14 were used for the calculation of the active tool length with $TCOABS. This still applies if $TC_CARR21 and $TC_CARR22 do not refer to rotary axes. If $TC_CARR21 or $TC_CARR22 contains a reference to a rotary axis in the channel, the axis position of the relevant axis at the start of the current block is used as the angle, rather than the entry in $TC_CARR13 or $TC_CARR14.

A mixed operating mode is permissible, i.e. the angles can be determined from the entry in the system variables $TC_CARR13 or $TC_CARR14 for one axis, and from the position of a channel axis for the other.

This makes it possible for machines, on which the axes used to set the toolholder with orientation capability are known within the NC, to access their position directly, whereas it was previously necessary, for example, to read system variable $AA_IM[axis] and write the result of the read operation to $TC_CARR13/14. In particular, this removes the implicit preprocessing stop when reading the axis positions.

MD20180

The rotary axis position is used with its programmed or calculated value, when the machine data:

MD20180 $MC_TOCARR_ROT_ANGLE_INCR[i] = 0 (rotary axis increment of the tool carrier that can be oriented)

If the machine data is not zero however, the position used is the nearest grid point obtained for a suitable integer value n from the equation:

\[ \phi = $MC_TOCARR_ROT_ANGLE_OFFSET[i] + n \times $MC_TOCARR_ROT_ANGLE_INCR[i] \]

This functionality is required if the rotary axes need to be indexed and cannot, therefore, assume freely-defined positions (e.g. with Hirth tooth systems). System variable $P_TCANG[i] delivers the approximated valued and system variable $P_TCDIFF[i] the difference between the exact and the approximated value.

Frame orientation TCOFR

With TCOFR (determination of the angle from the orientation defined by an active frame), the increments are scaled after determination of the angle from the active frame rotation. If the requested orientation is not possible due to the machine kinematic, the machining is aborted with an alarm. This also applies if the target orientation is very close to an achievable orientation. In particular the alarm in such situations cannot be prevented through the angle approximation.

TCARR frame offset

A frame offset as a result of a toolholder change becomes effective immediately on selection of TCARR= . . . . A change in the tool length, on the other hand, only becomes effective immediately if a tool is active.
TCOFR/TCOABS frame rotation

A frame rotation does not take place on activation and a rotation which is already active is not changed. As in case T (only the tool can be rotated), the position of the rotary axes used for the calculation is dependent on the G code TCOFR/TCOABS and determined from the rotation component of an active frame or from the entries $TC_CARRn.

Activation of a frame changes the position in the workpiece coordinate system accordingly, without compensating motion by the machine itself. The ratios are shown in the figure below:

![Diagram showing frame rotation and coordinate system](image)

Figure 18-43 Zero offset on activation of a rotary table with TCARR

Example

On the machine in the figure, the rotary axis of the table is pointing in the positive Y direction. The table is rotated by +45 degrees. PAROT defines a frame, which similarly describes a rotation of 45 degrees about the Y axis. The coordinate system is not rotated relative to the actual environment (marked in the figure with "Position of the coordinate system after TCARR"), but is rotated by -45 degrees relative to the defined coordinate system (position after PAROT). If this coordinate system is defined with ROT Y-45, for example, and if the toolholder is then selected with active TCOFR, an angle of +45 degrees will be determined for the rotary axis of the toolholder.

Rotary table

With rotary tables (kinematic types P and M), activation with TCARR similarly does not lead to an immediate rotation of the coordinate system (see figure), i.e. even though the zero point of the coordinate system is offset relative to the machine, while remaining fixed relative to the zero point of the workpiece, the orientation remains unchanged in space.
Activation of kinematic types P and M

With kinematics of type P and M the selection of a toolholder activates an additive frame (table offset of the toolholder with orientation capability), which takes into account the zero point offset as a result of the rotation of the table.

The zero offset can be written to a dedicated system frame $P\_PARTFR. For this, the bit 2 must be set in the machine data:

MD28082 $MC\_MM\_SYSTEM\_FRAME\_MASK (system frames (SRAM))

The basic frame identified by following machine data is then no longer required for the zero offset:

MD20184 $MC\_TOCARR\_BASE\_FRAME\_NUMBER (number of the basic frames for taking the table offset)

Activation of kinematic type M

With kinematics of type M (tool and table are each rotary around one axis), the activation of a toolholder with TCARR simultaneously produces a corresponding change in the effective tool length (if a tool is active) and the zero offset.

Rotations

Depending on the machining task, it is necessary to take into account not only a zero offset (whether as frame or as tool length) when using a rotary toolholder or table, but also a rotation. However, the activation of a toolholder with orientation capability never leads directly to a rotation of the coordinate system.

TOROT

If only the tool can be rotated, a frame whose Z axis points in the direction of the tool can be defined with TOFRAME or TOROT.

PAROT

If the coordinate system needs to be fixed relative to the workpiece, i.e. not only offset relative to the original position but also rotated according to the rotation of the table, then PAROT can be used to activate such a rotation in a similar manner to the situation with a rotary tool.

With PAROT, the translations, scalings and mirrorings in the active frame are retained, but the rotation component is rotated by the rotation component of a toolholder with orientation capability corresponding to the table.

PAROT and TOROT take into account the overall change in orientation in cases where the table or the tool are oriented with two rotary axes. With mixed kinematics only the corresponding component caused by a rotary axis is considered. It is thus possible, for example, when using TOROT, to rotate a workpiece such that an oblique plane lies parallel to the XY plane fixed in space, whereby rotation of the tool must be taken into account in machining where any holes to be drilled, for example, are not perpendicular to this plane.
Language command **PAROT** is not rejected if no toolholder with orientation capability is active. This causes no changes in the programmed frame.

**Note**

For further information about the **TCARR** and **TOROT** as well as **PAROT** functions with regard to channel-specific system frames, see Section "K2: Axis Types, Coordinate Systems, Frames (Page 661)".

### 18.6.5 Procedure when using toolholders with orientation capability

#### Creating a toolholder

The number of available toolholder data sets in the NCK is defined with machine data:

**MD18088 $MN_MM_NUM_TOOL_CARRIER** (maximum number of definable tool carriers)

1. The value is calculated as follows:

   \[
   MD18088 = \text{"Number of TO units" \times \"Number of toolholder data sets of a TO unit"} \\
   MD18088/\text{"number of TO units"}
   \]

   is permanently allocated to each TO unit.

**Note**

For further explanations on the definition and assignment of a TO unit by machine data:

**MD28085 $MC_MM_LINK_TOA_UNIT** (assignment of a TO unit to a channel (SRAM))

**References:**

Function Manual, Extended Functions; Memory Configuration (S7)

2. Zero setting of toolholder data:

   You can use the command $TC_CARR1[0] = 0 to zero all values of all data sets.

   Individual toolholder data sets can be deleted selectively with the NC command **DELT**C or the PI service **_N_DELTCAR**.

3. Accessing the data of a toolholder:

   - **Part program**
     
     → $TC_CARRn[m] = value
     
     This describes the previous value of the system variables n for toolholder m with the new value "value".
     
     → value = $TC_CARRn[m]
     
     With "def real value" - the parameters of a toolholder m can be read if they have already been defined (e.g. set MD18088). Otherwise, an alarm is signaled.

   - **OPI interface**
     
     The parameters of a toolholder with orientation capability can be read and written with the NCKHMI (OPI) variable services using system variable $P_TCANG[<n>].

4. Data backup:

   The system variables specified above are saved as part of the general NCK data backup.
Selecting the toolholder

A toolholder with number m is selected with the TCARR = m NC program command (TCARR: Tool Carrier).

TCARR = 0 deselects an active toolholder.

New tool or new toolholder

When a new tool is activated, it is always treated as if it was mounted on the active toolholder.

A new toolholder is activated immediately when it is programmed. It is not necessary to change tools or reprogram the active tool. The toolholder (number) and tool (number) are independent and can be used in any combination.

Toolholder from G code of group 42

Absolute tool orientation TCOABS (Tool Carrier Orientation ABSolute):

Tool orientation is determined explicitly if the corresponding values are entered in system variable $TC_CARR13 or $TC_CARR14 and G code TCOABS is activated in G-code group 42.

Frame tool orientation TCOFR (Tool Carrier Orientation FRame):

Tool orientation can also be determined automatically from the current orientation of an active frame when selecting a tool, if one of the following G codes is active in G-code group 42 when the toolholder is selected:

- TCOFR or TCOFRZ
  The toolholder with orientation capability is set so that the tool points in the Z direction.

- TCOFRX
  The toolholder with orientation capability is set so that the tool points in the X direction.

- TCOFRY
  The toolholder with orientation capability is set so that the tool points in the Y direction.

The effect of TCOFR is such that, when machining on an inclined surface, tool compensations are considered implicitly as if the tool were standing vertically on the surface.

Note

The tool orientation is not bound strictly to the frame orientation. When a frame is active and G code TCOABS is active, you can select a tool, whereby the orientation of the tool is independent of the orientation of the active frame.

Following tool selection, you can change the frame, which does not affect the components of tool length compensation. It is then no longer certain that the tool is positioned perpendicular to the machining plane. You should therefore first check that the intended tool orientation is maintained on an inclined surface.

When TCOFR, etc., is active, the tool orientation used in the tool length calculation is always determined from the active frame each time the toolholder is changed.
Toolholder from G code of group 53

The G codes of group 53 (TOFRAME, TOROT, etc.) can be used to define a frame such that an axis direction (Z, Y or X) in this frame is equal to the current tool orientation.

The G code of group 6 (G17 - G19), which is active at the time TOFRAME is called, determines the tool orientation.

If no toolholder is active, or if a toolholder is active but does not cause the tool orientation to change, the Z direction in the new frame is:

- The same as the old Z direction with G17.
- The same as the old Y direction with G18.
- The same as the old X direction with G19.

These directions are modified accordingly for rotating toolholders. The same applies to the new X and Y directions.

Instead of TOFRAME or TOROT, one of the G codes TOFRAMEX, TOFRAMEY, TOROTX, or TOROTY can be used. The meanings of the axes are interchanged accordingly.

Group change

Changing the G code from group 42 (TCOABS, TCOFR, etc.) causes recalculation of the tool length components.

The (programmed) angles of rotation stored in the toolholder data are not affected, with the result that the angles originally stored in the toolholder data are reactivated on a change from TCOFR to TCOABS.

Read rotary angle (α1 or α2):

The angles currently used to calculate the orientation can be read via system variable $P_TCANG[n]$ where n = 1 or n = 2.

If two permissible solutions (i.e., a second valid pair of angles) are available for a particular orientation, the values can be accessed with $P_TCANG[3]$ or $P_TCANG[4]$. The number of valid solutions 0 to 2 can be read with $P_TCSOL$.

Tool radius compensation with CUT2D or CUT3DFS:

The current tool orientation is included in the tool radius compensation if either CUT2D or CUT3DFS is active in G-code group 22 (tool compensation type).

For nonrotating toolholders, the behavior depends solely on the active plane of G code group 6 (G17 - G19) and is, therefore, identical to the previous behavior.

All other tool compensation types:

The behavior for all other tool compensation types is unchanged.

For CUT2DF and CUT3DFF in particular, the compensation plane used for TRC is determined from the active frame, independent of the current tool orientation. Allowance is made for the active plane (G17 - G19) and the behavior is, therefore, the same as before.

The two remaining G codes of group 22, CUT3DC and CUT3DF, are not affected by the toolholder functionality because the tool orientation information in these cases is made available by the active kinematic transformation.
**Two rotary axes**

Two general solutions exist for two rotary axes. The control itself chooses these two solution pairs such that the orientation angles resulting from the frame are as close as possible to the specified angles.

The two following options are available for specifying the angles:

1. If $TC_CARR21$ or $TC_CARR22$ contains a reference to a rotary axis, the position of this axis at the start of the block in which the toolholder is activated is used to specify the angle.
2. If $TC_CARR21$ or $TC_CARR22$ does not contain a reference to a rotary axis, the values contained in $TC_CARR13$ or $TC_CARR14$ are used.

**Example**

The control first calculates an angle of 10 degrees for one axis. The specified angle is 750 degrees. 720 degrees (= 2 * 360 degrees) are then added to the initial angle, resulting in a final angle of 730 degrees.

**Rotary axis offset**

Rotary axis offsets can be specified with system variables $TC_CARR24$ and $TC_CARR25$. A value not equal to zero in one of these parameters means that the initial state of the associated rotary axis is the position specified by the parameter (and not position zero). All angle specifications then refer to the coordinate system displaced by this value.

When the machining plane is changed (G17 - G19), only the tool length components of the active tool are interchanged. The components of the toolholder are not interchanged. The resulting tool length vector is then rotated in accordance with the current toolholder and, if necessary, modified by the offsets belonging to the toolholder.

The two toolholder angles $\alpha_1$ and $\alpha_2$ are determined using a frame, independent of the active plane currently selected (G17 - G19).

**Limit values**

Limit angles (software limits) can be specified for each rotary axis in the system variable set ($TC_CARR30$ to $TC_CARR33$) used to describe the toolholder with orientation capability. These limits are not evaluated if both the minimum and maximum value is zero.

If at least one of the two limits is not equal to zero, the system checks whether the previously calculated solution is within the permissible limits. If this is not the case, an initial attempt is made to reach a valid setting by adding or subtracting multiples of 360 degrees to or from the invalid axis position. If this is impossible and two different solutions exist, the first solution is discarded and the second solution is used. The second solution is treated the same as the first with reference to the axis limits.

If the first solution is discarded and the second used instead, the contents of $P_TCANG[1/2]$ and $P_TCANG[3/4]$ are swapped, hence the solution actually used is also stored in $P_TCANG[1/2]$ in this case.

The axis limits are monitored even if the axis angle is specified instead of being calculated. This is the case if $TCOABS$ is active when a toolholder with orientation capability is activated.
18.6.6 Programming

Selecting the toolholder
A toolholder is selected with the number \( m \) of the toolholder with:

\[ TCARR = m \]

Access to toolholder data blocks
The following access is possible from the part program:

The current value of the parameter \( n \) for the tool holder \( m \) is written with the new "value" with:

\[ $TC_CARRn[m] = \text{value} \]

The parameters of a tool holder \( m \) can, as far as the toolholder data set is already defined, read with:

\[ \text{value} = $TC_CARRn[m] \] (Value must be a REAL variable)

The toolholder data set number must lie in the range, which is defined by the machine data:

MD18088 $MN_MM_NUM_TOOL_CARRIER (Total number of toolholder data sets that can be defined)

This number of toolholder data sets, divided by the number of active channels, can be defined for a channel.

Exception:
If settings, which deviate from the standard, are selected via the machine data:

MD28085 $MC_MM_LINK_TOA_UNIT (Assignment of TO unit to a channel).

Canceling all toolholder data blocks
All values of all toolholder data sets can be deleted from within the part program using one command.

\[ $TC_CARR1[0] = 0 \]

Values not set by the user are preset to 0.

Activation
A toolholder becomes active when both a toolholder and a tool have been activated. The selection of the toolholder alone has no effect. The effect of selecting a toolholder depends on the G code \( TCOABS / TCOFR \) (modal G-code group for toolholders).

Changing the G code in the \( TCOABS / TCOFR \) group causes recalculation of the tool length components when the toolholder is active. With \( TCOABS \), the values stored in the toolholder data for both angles of rotation \( \alpha_1 \) and \( \alpha_2 \) are used to determine the tool orientation.

With \( TCOFR \), the two angles are determined from the current frame. The values stored in the toolholder data are not changed, however. These are also used to resolve the ambiguity that
can result when the angle of rotation is calculated from one frame. Here, the angle that deviates least from the programmed angle is selected from the various possible angles.

Note
For more explanations on the programming of tool compensations with toolholder kinematic and for the system variables see:

References:
Programming Manual, Job Planning

18.6.7 Supplementary conditions and control system response for orientation

Full orientation
For a given data set that describes a certain kinematic, all the conceivable special orientations can only be displayed when the following conditions are satisfied:

- The two vectors \( v_1 \) and \( v_2 \) that describe the rotary axes must also be defined (i.e. both vectors must not be equal to zero).
- The two vectors \( v_1 \) and \( v_2 \) must be perpendicular to each other.
- The tool orientation must be perpendicular to the second rotary axis.

Non-defined orientation
If these conditions are not satisfied and an orientation that cannot be achieved by an active frame is requested with \( \text{TCOFR} \), an alarm is output.

Vector/angle of rotation dependencies
If vector \( v_1 \) or \( v_2 \), which describes the direction of a rotary axis, is set to zero, the associated angle of rotation \( \alpha_1 \) or \( \alpha_2 \) must also be set to zero. Otherwise, an alarm is produced. The alarm is not output until the toolholder is activated, i.e. when the toolholder is changed.

Tool fine compensation combined with orientation
Tool fine compensations and toolholders cannot be combined. The activation of tool fine compensation when a toolholder is active, and vice versa the activation of the toolholder when tool fine compensation is active, produces an alarm.

Automatic toolholder selection, \( \text{RESET} \)
For \( \text{RESET} \) or at program start, a toolholder can be selected automatically via the machine data:

\[ \text{MD20126} \ \$\text{MC\_TOOL\_CARRIER\_RESET\_VALUE} \] (Active toolholder at \( \text{RESET} \))

It is handled similar to the controlled selection of a tool via the machine data:

\[ \text{MD20120} \ \$\text{MC\_TOOL\_RESET\_VALUE} \] (Tool length compensation Power up (\( \text{RESET/TP-End} \)))
The behavior at **RESET** or at program start is controlled as in the case of tool selection via the same bit 6 in the machine data:

**MD20110 $MC_RESET_MODE_MASK** (definition of initial control settings after RESET/TP-End)

Or:

**MD20112 $MC_START_MODE_MASK** (definition of initial control system settings at NC-START)

For further information, see Section "K1: Mode group, channel, program operation, reset response (Page 455)"

**SW 6.3 and higher**

If **TCOABS** was active for the last selection before reset, the behavior is unchanged compared to previous versions. A different active G code causes the toolholder with orientation capability to be activated with the frame that was active before the last reset. Modified toolholder data ($TC_CARR...) are also considered. If these data are unchanged, the toolholder is activated in exactly the same state as before reset. If the toolholder data were changed after the toolholder selection before reset, selection corresponding to the last frame is not always possible. In this case, the toolholder with orientation capability is selected according to the G-Code (group 42) values valid at this time and the active frame.

**MD22530 output of auxiliary functions to PLC**

That, optionally, a constant or an M code is output when the toolholder is selected, whose number of the code is derived from the toolholder number. Can be set with the machine data:

**MD22530 $MC_TOCARR_CHANGE_M_CODE** (M code at toolholder change)

For further information, see Section "K1: Mode group, channel, program operation, reset response (Page 455)"

**Toolholder kinematics**

The following supplementary conditions must be met for toolholder kinematics:

- Tool orientation in initial state, both angles $\alpha_1$ and $\alpha_2$ zero, as per default setting, even if:
  - G17 parallel to Z
  - G18 Parallel to Y
  - G19 parallel to Z

- A permissible position in terms of the axis limits must be achievable.

- For any possible orientation to be set, the two rotary axes must be perpendicular to each other.
  For machines, on which the table is rotated by both axes, the tool orientation must also be perpendicular to the first rotary axis.
  For machines with mixed kinematics, the tool orientation must be perpendicular to the axis which rotates the tool, i.e. also the first rotary axis.
The following applies to orientations specified in a frame:

- The orientation specified in a frame must be achievable with the defined toolholder kinematics, otherwise an alarm is output. This situation can occur if the two rotary axes required to define the kinematics are not perpendicular to each other. This applies if fewer than two rotary axes are defined and is the case:
  - With **kinematic type T with rotary tool**, if the tool axis, which defines the tool direction, is not perpendicular to the second axis.
  - With **kinematic types M and P with rotary workpiece**, if the tool axis, which defines the tool direction, is not perpendicular to the first axis.

- Rotary axes, which require a frame with a defined tool orientation in order to reach a specific position, are only determined unambiguously in the case of one rotary axis. Two general solutions exist for two rotary axes.

- In all cases where ambiguities may arise, it is particularly important that the approximate angles expected from the frame are stored in the tool data, and that the rotary axes are in the vicinity of the expected positions.

**Response with ASUP, REPOS**

The toolholder can be changed in an asynchronous subprogram (ASUB). When the interrupted program is resumed with **REPOS**, the approach motion of the new toolholder is taken into account and the program continues with this motion. The treatment here is analogous to tool change in an ASUB.

For further information, see Section "**K1: Mode group, channel, program operation, reset response** (Page 455)".

### 18.7 Cutting edge data modification for tools that can be rotated

#### 18.7.1 Function

Using the function "cutting data modification for rotatable tools", the changed geometrical relationships, that are obtained relative to the workpiece being machined when rotating tools (predominantly turning tools, but also drilling and milling tools) can be taken into account.

#### 18.7.2 Determination of angle of rotation

The current rotation of the tool is always determined from a currently active, orientable toolholder (see Section "**Toolholder with orientation capability** (Page 1442)").

The angle of rotation of the toolholder with orientation capability is normally (but not necessarily) defined with the \texttt{TCOFR} command from an active frame. This method can be used to define the tool orientation independently of the actual kinematics with which the tool is rotated, identically in each case with the help of two angles.
The two machine-independent orientation angles $\beta$ (Beta) and $\gamma$ (Gamma) are used to define the tool rotation. $\beta$ is the angle of rotation and the applicate (typically a B axis in G18) and $\gamma$ a rotation around the ordinate (typically a C axis in G18). The rotation is first executed around $Y$, finally around $\beta$, i.e. the $y$ axis is rotated by the $\beta$ axis:

18.7.3 Cutting edge position, cut direction and angle for rotary tools

Turning tools

Turning tools means the following tools whose tool type ($STC\_DP1$) has values in the range of 500 to 599. Grinding tools (tool types 400 to 499) are equivalent to turning tools.

Tools are treated independently of tool type such as turning tools if:

$SD42950 \; SC\_TOOL\_LENGTH\_TYPE = 2$
Cutting edge position and cut direction

Turning tools are limited by their main and secondary cutting edges. The tool parameter "Cutting edge position" is defined via the position of these two cutting edges relative to the coordinate axes. The ratios are displayed with diagram in the following figure:

![Diagram showing cutting edge position and cut direction for turning tools.]

The values 1 to 4 characterize the cases in which both cutting edges lie in the same quadrant; the values 5 to 8 characterize the cases in which both cutting edges lie in neighboring quadrants or there is a coordinate axis between the two cutting edges. The cutting edge position is stored in the tool parameter $TC\_DP2$.

A cut direction can be defined for each turning tool. It is stored in the tool parameter $TC\_DP11$. It has values between 1 and 4, and it characterizes a positive or negative direction of the coordinate axes:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ordinate -</td>
</tr>
<tr>
<td>2</td>
<td>Ordinate +</td>
</tr>
<tr>
<td>3</td>
<td>Abscissa -</td>
</tr>
<tr>
<td>4</td>
<td>Abscissa +</td>
</tr>
</tbody>
</table>

Two different cut directions can be assigned to each cutting edge position:

<table>
<thead>
<tr>
<th>Cutting edge position:</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>
</tr>
</thead>
<tbody>
<tr>
<td>Cut direction:</td>
<td>2, 4</td>
<td>2, 3</td>
<td>1, 3</td>
<td>1, 4</td>
<td>1, 2</td>
<td>3, 4</td>
<td>1, 2</td>
<td>3, 4</td>
</tr>
</tbody>
</table>
**Holder angle and clearance angle**

The following figure depicts the two angles (holder angle and clearance angle) of a turning tool with cutting edge position 3, that are necessary for describing the geometry of the tool cutting edge. The cut direction in this example is 3, i.e. it denotes the negative Z direction (abscissa direction for G18).

![Figure 18-45 Angle and cut direction for a turning tool with cutting edge position 3](image)

The cut direction specifies the reference direction of the holder angle. The clearance angle is the angle measured between the inverse cut direction and the adjacent cutting edge (positive). Holder angle and clearance angle are stored in the tool parameters $TC\_DP10$ or $TC\_DP24$.

**Note**

Cut direction and tool angle are relevant only in the cutting edge positions 1 to 8.
18.7.4 Modifications during the rotation of turning tools

Tool orientation

Unlike milling tools, turning tools are not rotation-symmetric. This means that normally 3 degrees of freedom or three rotary axes are required to describe the tool orientation. The concrete kinematics therefore, is independent of the machine only to the extent the desired orientation can be set. If necessary, the third degree of freedom can be substituted by a rotation of the tool coordinate system.

Note

The division of the orientation into one component created by the toolholder with orientation capability and a second component achieved via a rotation of the coordinate system is the responsibility of the application. The control does not provide any further functionality in this regard.

Shape of cutting edge

If a turning tool turns by an angle against the machining plane (i.e. around an axis in the machining plane, typically a C axis) that is not a multiple of 180°, then the configuration of the (circular) tool cutting edge in the machining plane becomes an ellipse. It is assumed that the deviations from the circular form arising on account of such rotations is so insignificant that they can be ignored (tilt angle < 5°), i.e. the control always ignores the tool orientation and assumes a circular cutting edge.

This also means that with reference to the active plane, the control accepts only a rotation by 180° as a setting deviating from the initial position. This limitation is valid for the shape of cutting edge only. The tool lengths are always considered correctly in random spatial rotations.

Cutting edge position, cut direction and angle

A rotation by 180° around an axis in the machining plane means that while using the tool at the same position, the spindle rotation direction with reference to the use of the unturned tool must be inverted.

Cut direction and cutting edge position are also not modified like the cutting edge reference point (see below) if the tool is rotated from the plane by +/- 90° (with a tolerance of app. 1°) because then the configuration of the cutting edge is not defined in the current plane.

If the tool rotates in the plane (rotation around an axis vertical to the machining place or around the Y axis for G18), the cutting edge position is determined from the resulting angle for the clearance and holder angles. If these two angles are not specified for the tool (i.e. $TC_DP10 and $TC_DP24 are both zero), then the new cutting edge position is determined from the turning angle alone. The cutting edge position changes only in 90° steps, i.e. the cutting edge position remains independent of the initial state either in the value range 1 to 4 or 5 to 8. The new cutting edge position is then determined exclusively from the angle of rotation if the specified values for holder angle and clearance angle are not allowed (negative values, resulting plate angle negative or more than 90°). Clearance angle and holder angle are not modified in all these cases.
Depending on the rotation, the cut direction is modified in such a way that the resulting clearance angle remains less than 90°. If the original cut direction and the original cutting edge position do not match, then the cut direction is not modified during rotation of the tool (see Section "Cutting edge position, cut direction and angle for rotary tools (Page 1469)").

The angle of rotation in the plane, as it was determined from the toolholder with orientation capability, is available in the system variable $P\_CUTMOD\_ANG$ or $AC\_CUTMOD\_ANG$. This angle is the original angle without any final rounding to multiples of 45° or 90°.

Limit cases

If, for a turning tool, the cutting edge position, cut direction, clearance and holder angles have valid values so that all cutting edge positions (1 to 8) are possible through suitable rotations in the plane, then the cutting edge positions 1 to 4 are preferred to cutting edge positions 5 to 8 in the cases in which one of the cutting edges (main or secondary cutting edge) is away from the coordinate axis by less than half the input increment ((0.0005° for an input specification of 3 decimal digits).

The following is applicable in all other cases (milling tools or turning tools without valid cutting edge parameters) in which rotation is possible only in 90° steps: If the amount of the rotation angle is smaller than 45° + 0.5 input increments (corresponds to 45.0005° for an input specification of three decimal places), the cutting edge position and cut direction are not changed, i.e. these cases are treated as rotations that are smaller than 45°. Rotations, the amount of which deviates from 180° by less than 45° + 0.5 input increments are treated identically as rotations in the range of 135° to 225°.

Cutting edge reference point

The cutting edge center point and the cutting edge reference point are defined for turning tools. The position of these two points relative to each other is defined by the cutting edge position.

The distance of the two points for cutting edge positions 1 to 4 is equal to √2 times the cutting edge radius; for cutting edge positions 5 to 8 it is equal to 1 times the cutting edge radius. In the first case, the cutting edge reference point relative to the cutting edge center point lies in the machining plane on a bisecting line, while in the second case it lies on a coordinate axis.

If you rotate the tool by a random angle around an axis vertical to the machining plane, the cutting edge reference point would also rotate if it had a fixed position relative to the tool. The above-mentioned condition (position on an axis or a bisecting axis) is not fulfilled in most cases. This is not desirable. Instead, the cutting edge reference point should always be modified in such a way that the distance vector between cutting edge reference point and cutting edge center point has one of the mentioned 8 directions. The cutting edge position must be modified for this if necessary.

The ratios are shown with examples in the figure below:
A tool with the cutting edge position 3, the clearance angle 22.5° and holder angle 112.5° is rotated. For rotations up to 22.5°, the cutting edge position is maintained, the position of the cutting edge reference point relative to the tool however, is compensated in such a way that the relative position of both points are maintained in the machining plane. For bigger rotations (up to 67.5°), the cutting edge position changes to value 8.

Note
As the cutting edge reference point is defined by the tool length vector, modifying the cutting edge reference point changes the effective tool length.

18.7.5 Cutting edge position for milling and tapping tools

Milling and tapping tools
Milling and tapping tools means the following tools whose tool type ($TC\_DP1$) has values in the range of 100 to 299.

Tools are treated independently of tool type such as milling and tapping tools if:

SD42950 $SC\_TOOL\_LENGTH\_TYPE = 1$

Length of cutting edge
A cutting edge position is also introduced for the so defined milling and tapping tools which is modified according to the following description, in case of rotations.

Any specified cutting edge position for tools that are not milling and tapping tools or turning tools according to the mentioned definitions, is not evaluated.
The cutting edge position of the tapping and milling tools is stored in tool parameter $TC\_DP2$ as in the case of turning tools. Based on the definition of the cutting edge position for turning tools, this parameter can assume the values 5 to 8. Here, the cutting edge position specifies the orientation (the direction of the rotation axis) of the tool:

<table>
<thead>
<tr>
<th>Length of cutting edge</th>
<th>Direction of rotation axis of tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Abscissa +</td>
</tr>
<tr>
<td>6</td>
<td>Ordinate +</td>
</tr>
<tr>
<td>7</td>
<td>Abscissa -</td>
</tr>
<tr>
<td>8</td>
<td>Ordinate -</td>
</tr>
</tbody>
</table>

Example

Figure 18-47  Milling tool with cutting edge position 7

18.7.6 Modifications during rotation of milling and tapping tools

The cutting edge position is recalculated appropriately during a rotation of a milling or tapping tool. Cut direction and tool angle (clearance angle or holder angle) are not defined for milling and tapping tools so that the change in cutting edge position is derived exclusively from the rotation. Thus, for milling and tapping tools, the cutting edge position always changes when the amount of rotation with reference to the zero setting is more than 45°.
18.7.7 Parameter assignment

Reaction to errors

Different fault conditions can occur during the activation of the "Cutting edge data modification for rotary tools" function (via explicit call with CUTMOD or through a tool selection).

For each of these possible fault conditions, one can define whether the error is to trigger an alarm output, whether such an alarm is only to be displayed (warning), or whether additionally the interpretation of the part program must be cancelled. The setting is done via the following machine data:

MD20125 $MC_CUTMOD_ERR

Two bits of the machine data are assigned to each fault condition:

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>No valid cut direction is defined for the active tool.</td>
<td>0</td>
<td>Alarm output for error &quot;Invalid cut direction&quot;</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Program stop for error &quot;Invalid cut direction&quot;</td>
</tr>
<tr>
<td>The cutting edge angle (clearance angle and holder angle) of the active tool are both zero.</td>
<td>2</td>
<td>Alarm output for error &quot;Not defined cutting edge angle&quot;</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Program stop for error &quot;Not defined cutting edge angle&quot;</td>
</tr>
<tr>
<td>The clearance angle of the active tool has an impermissible value (&lt; 0° or &gt; 180°).</td>
<td>4</td>
<td>Alarm output for error &quot;Invalid clearance angle&quot;</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Program stop for error &quot;Invalid clearance angle&quot;</td>
</tr>
<tr>
<td>The holder angle of the active tool has an impermissible value (&lt; 0° or &gt; 90°).</td>
<td>6</td>
<td>Alarm output for error &quot;Invalid holder angle&quot;</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Program stop for error &quot;Invalid holder angle&quot;</td>
</tr>
<tr>
<td>The plate angle of the active tool has an impermissible value (&lt; 0° or &gt; 90°).</td>
<td>8</td>
<td>Alarm output for error &quot;Invalid plate angle&quot;</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Program stop for error &quot;Invalid plate angle&quot;</td>
</tr>
<tr>
<td>The cutting edge position - holder angle combination of the active tool is not permitted (the holder angle must be ≤ 90° for cutting edge position 1 to 4; for cutting edge positions 5 to 8 it must be ≥ 90°).</td>
<td>10</td>
<td>Alarm output for error &quot;Invalid cutting edge position - holder angle combination&quot;</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Program stop for error &quot;Invalid cutting edge position - holder angle combination&quot;</td>
</tr>
<tr>
<td>Inadmissible rotation of the active tool (the tool was rotated from the active machining plane by ± 90° (with a tolerance of about 1°). Hence the cutting edge position is no longer defined in the machining plane.</td>
<td>12</td>
<td>Alarm output for error &quot;Invalid rotation&quot;</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>Program stop for error &quot;Invalid rotation&quot;</td>
</tr>
</tbody>
</table>

Response to POWER ON

The "cutting edge data modification for rotary tools" function (CUTMOD) is initialized automatically during POWER ON with the value stored in machine data:

MD20127 $MC_CUTMOD_INIT

If the value of this machine data is "-2", CUTMOD is set to the value that is set in machine data:

MD20126 $MC_TOOL_CARRIER_RESET_VALUE (Active toolholder at RESET)
18.7.8 Programming

The "Cutting edge data modification for rotary tools" function is activated with the CUTMOD command.

Syntax

CUTMOD=<value>

Meaning

CUTMOD Command to switch-in the function "cutting data modification for tools that can be rotated"

<value> The following values can be assigned to the CUTMOD command:

- 0 The function is deactivated.
   The values supplied from system variables $P_AD... are the same as the corresponding tool parameters.
- > 0 The function is activated if a toolholder that can be orientated with the specified number is active, i.e. the activation is linked to a specific toolholder that can be orientated.
   The values supplied from system variables $P_AD... may be modified with respect to the corresponding tool parameters depending on the active rotation.
   The deactivation of the designated toolholder that can be orientated temporarily deactivates the function; the activation of another toolholder that can be orientated permanently deactivates it. This is the reason that in the first case, the function is re-activated when again selecting the same toolholder that can be orientated; in the second case, a new selection is required - even if at a subsequent time, the toolholder that can be orientated is re-activated with the specified number.
   The function is not influenced by a reset.
- -1 The function is always activated if a toolholder that can be orientated is active.
   When changing the toolholder or when de-selecting it and a subsequent new selection, CUTMOD does not have to be set again.
- -2 The function is always activated if a toolholder that can be orientated is active whose number is the same as the currently active toolholder that can be orientated.
   If a toolholder that can be orientated is not active, then this has the same significance as CUTMOD=0.
   If a toolholder that can be orientated is active, then this has the same significance as when directly specifying the actual toolholder number.
- < -2 Values less than 2 are ignored, i.e. this case is treated as if CUTMOD was not programmed.

Note:
This value range should not be used as it is reserved for possible subsequent expansions.
Note

SD42984 $SC_CUTDIRMOD

The function that can be activated using the CUTMOD command replaces the function that can be activated using the setting data SD42984 $SC_CUTDIRMOD. However, this function remains available unchanged, because it doesn't make sense to use both functions in parallel, it can only be activated if CUTMOD is equal to zero.

Effectiveness of the modified cutting data

The modified tool nose position and the modified tool nose reference point are immediately effective when programming, even for a tool that is already active. A tool does not have to be re-selected for this purpose.

Influence of the active machining plane

To determine modified tool nose position, cutting direction and holder or clearance angle, the evaluation of the cutting edge in the active plane (G17 - G19) is decisive.

However, if setting data SD42940 $SC_TOOL_LENGTH_CONST (change of the tool length component when selecting the plane), contains a valid value not equal to zero (plus or minus 17, 18 or 19), then its contents define the plane in which the relevant quantities are evaluated.
The following system variables are available:

<table>
<thead>
<tr>
<th>System variables</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_CUTMOD_ANG / $AC_CUTMOD_ANG</td>
<td>Supplies the (non-rounded) angle in the active machining plane that was used as basis for the modification of the cutting data (tool nose position, cut direction, clearance angle and holder angle) for the functions activated using CUTMOD and/or $SC_CUTDIRMOD. $P_CUTMOD_ANG refers to the actual state in the preprocessing, $AC_CUTMOD_ANG to the actual main run block.</td>
</tr>
<tr>
<td>$P_CUTMOD / $AC_CUTMOD</td>
<td>Reads the currently valid value that was last programmed using the command CUTMOD (number of the toolholder that should be activated for the cutting data modification). If the last programmed CUTMOD value = -2 (activation with the currently active toolholder that can be orientated), then the value -2 is not returned in $P_CUTMOD, but the number of the active toolholder that can be orientated at the time of programming. $P_CUTMOD refers to the actual state in the preprocessing, $AC_CUTMOD to the actual main run block.</td>
</tr>
<tr>
<td>$P_CUT_INV / $AC_CUT_INV</td>
<td>Supplies the value TRUE if the tool is rotated so that the spindle direction of rotation must be inverted. To do this, the following four conditions must be fulfilled in the block to which the read operations refer: 1. If a turning or grinding tool is active (tool types 400 to 599 and/or SD42950 $SC_TOOL_LENGTH_TYPE = 2). 2. If the cutting influence was activated using the language command CUTMOD. 3. If a toolholder that can be orientated is active, which was designated using the numerical value of CUTMOD. 4. If the toolholder that can be orientated rotates the tool around an axis in the machining plane (this is typically the C axis) so that the resulting perpendicular of the tool cutting edge is rotated with respect to the initial position by more than 90° (typically 180°). The contents of the variable is FALSE if at least one of the specified four conditions is not fulfilled. For tools whose tool nose position is not defined, the value of the variable is always FALSE. $P_CUT_INV refers to the actual state in the preprocessing and $AC_CUT_INV to the actual main run block.</td>
</tr>
</tbody>
</table>

All main run variables ($AC_CUTMOD_ANG, $AC_CUTMOD and $AC_CUT_INV) can be read in synchronized actions. A read access operation from the preprocessing generates a preprocessing stop.

**Modified cutting data**

If a tool rotation is active, the modified data is made available in the following system variables:

<table>
<thead>
<tr>
<th>System variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_AD[2]</td>
<td>Cutting edge position</td>
</tr>
<tr>
<td>$P_AD[10]</td>
<td>Holder angle</td>
</tr>
</tbody>
</table>
### 18.7 Cutting edge data modification for tools that can be rotated

<table>
<thead>
<tr>
<th>System variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_AD[24]$</td>
<td>Clearance angle</td>
</tr>
</tbody>
</table>

**Note**

The data is always modified with respect to the corresponding tool parameters ($TC\_DP2[\ldots, \ldots]$ etc.) if the function "cutting data modification for rotatable tools" was activated using the command `CUTMOD` and a toolholder that can be orientated, which causes a rotation, is activated.

---

**18.7.9 Example**

The following example refers to a tool with tool nose position 3 and a toolholder that can be orientated, which can rotate the tool around the B axis.

The numerical values in the comments specify the end of block positions in the machine coordinates (MCS) in the sequence X, Y, Z.

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 $TC_DP1[1,1]=500</td>
<td></td>
</tr>
<tr>
<td>N20 $TC_DP2[1,1]=3 ; Length of cutting edge</td>
<td></td>
</tr>
<tr>
<td>N30 $TC_DP3[1,1]=12</td>
<td></td>
</tr>
<tr>
<td>N40 $TC_DP4[1,1]=1</td>
<td></td>
</tr>
<tr>
<td>N50 $TC_DP6[1,1]=6</td>
<td></td>
</tr>
<tr>
<td>N60 $TC_DP10[1,1]=110 ; Holder angle</td>
<td></td>
</tr>
<tr>
<td>N70 $TC_DP11[1,1]=3 ; Cut direction</td>
<td></td>
</tr>
<tr>
<td>N80 $TC_DP24[1,1]=25 ; Clearance angle</td>
<td></td>
</tr>
<tr>
<td>N110 $TC_CARR13[2]=0</td>
<td></td>
</tr>
<tr>
<td>N120 $TC_CARR14[2]=0</td>
<td></td>
</tr>
<tr>
<td>N130 $TC_CARR21[2]=X</td>
<td></td>
</tr>
<tr>
<td>N140 $TC_CARR22[2]=X</td>
<td></td>
</tr>
<tr>
<td>N150 $TC_CARR23[2]=&quot;M&quot;</td>
<td></td>
</tr>
<tr>
<td>N160 TCOABS CUTMOD=0</td>
<td></td>
</tr>
<tr>
<td>N170 G18 T1 D1 TCARR=2 X Y Z</td>
<td></td>
</tr>
<tr>
<td>N180 X0 Y0 Z0 F10000 ; 12.000 0.000 1.000</td>
<td></td>
</tr>
<tr>
<td>N190 $TC_CARR13[2]=30</td>
<td></td>
</tr>
<tr>
<td>N200 TCARR=2</td>
<td></td>
</tr>
<tr>
<td>N210 X0 Y0 Z0 ; 10.892 0.000 -5.134</td>
<td></td>
</tr>
<tr>
<td>N220 G42 Z=-10 ; 8.696 0.000 -17.330</td>
<td></td>
</tr>
<tr>
<td>Program code</td>
<td>Comment</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>N230 Z–20</td>
<td>; 8.696 0.000 –21.330</td>
</tr>
<tr>
<td>N240 X10</td>
<td>; 12.696 0.000 –21.330</td>
</tr>
<tr>
<td>N250 G40 X20 Z0</td>
<td>; 30.892 0.000 –5.134</td>
</tr>
<tr>
<td>N260 CUTMOD=2 X0 Y0 Z0</td>
<td>; 8.696 0.000 –7.330</td>
</tr>
<tr>
<td>N270 G42 Z–10</td>
<td>; 8.696 0.000 –17.330</td>
</tr>
<tr>
<td>N280 Z–20</td>
<td>; 8.696 0.000 –21.330</td>
</tr>
<tr>
<td>N290 X10</td>
<td>; 12.696 0.000 –21.330</td>
</tr>
<tr>
<td>N300 G40 X20 Z0</td>
<td>; 28.696 0.000 –7.330</td>
</tr>
<tr>
<td>N310 M30</td>
<td></td>
</tr>
</tbody>
</table>

Explanations:

In block N180, initially the tool is selected for CUTMOD=0 and non-rotated toolholders that can be orientated. As all offset vectors of the toolholder that can be orientated are 0, the position that corresponds to the tool lengths specified in $STC_DP3[1,1]$ and $STC_DP4[1,1]$ is approached.

The toolholder that can be orientated with a rotation of 30° around the B axis is activated in block N200. As the tool nose position is not modified due to CUTMOD=0, the old tool nose reference point is decisive just as before. This is the reason that in block N210 the position is approached, which keeps the old tool nose reference point at the zero (i.e. the vector (1, 12) is rotated through 30° in the Z/X plane).

In block N260, contrary to block N200 CUTMOD=2 is effective. As a result of the rotation of the toolholder that can be orientated, the modified tool nose position becomes 8. The consequence of this is also the different axis positions.

The tool radius compensation (TRC) is activated in blocks N220 and/or N270. The different tool nose positions in both program sections has no effect on the end positions of the blocks in which the TRC is active; the corresponding positions are therefore identical. The different tool nose positions only become effective again in the deselect blocks N260 and/or N300.

### 18.8 Incrementally programmed compensation values

#### 18.8.1 G91 extension

Requirements

Incremental programming with G91 is defined such that the compensation value is traversed additively to the incrementally programmed value when a tool compensation is selected.
Applications

For applications such as scratching, it is necessary only to traverse the path programmed in the incremental coordinates. The activated tool compensation is not traversed.

Sequence

Selection of a tool compensation with incremental programming

- Scratch workpiece with tool tip.
- Save the actual position in the basic frame (set actual value) after reducing it by the tool compensation.
- Traverse incrementally from the zero position.

Activation

It is possible to set whether a changed tool length is traversed with FRAME and incremental programming of an axis, or whether only the programmed path is traversed with the setting data:

SD42442 $SC_TOOL_OFFSET_INCR_PROG (tool length compensations)

Zero offset / frames G91

It is possible to set whether a zero offset is traversed as standard with value = 1 with FRAME and incremental programming of an axis, or whether only the programmed path is traversed with value = 0 with the setting data:

SD42440 $SC_FRAME_OFFSET_INCR_PROG (zero offset in frames)

For further information, see Section "K2: Axis Types, Coordinate Systems, Frames (Page 661)".

Supplementary condition

If the behavior is set such that the offset remains active even after the end of the program and

MD20110 $MC_RESET_MODE_MASK, bit6=1 (specification of the controller initial setting after reset / TP end)

and if an incremental path is programmed in the first part program block, the compensation is always traversed additively to the programmed path.

Note

With this configuration, part programs must always begin with absolute programming.
18.8.2 Machining in direction of tool orientation

Typical application

On machines with toolholders with orientation capability, traversing should take place in the tool direction (typically, when drilling) without activating a frame (e.g., using TOFRAME or TOROT), on which one of the axes points in the direction of the tool.

This is also true of machines on which a frame defining the oblique plane is active during oblique machining operations, but the tool cannot be set exactly perpendicular because an indexed toolholder (Hirth tooth system) is restricting the setting of the tool orientation.

In these cases it is then necessary - contrary to the motion actually requested perpendicular to the plane - to drill in the tool direction, as the drill would otherwise not be guided in the direction of its longitudinal axis, which, among other things, would lead to breaking of the drill.

MOVT

The end point of such a motion is programmed with MOVТ= ... . The programmed value is effective incrementally in the tool direction as standard. The positive direction is defined from the tool tip to the toolholder. The content of MOVТ is thus generally negative for the feed motion (when drilling), and positive for the retraction motion. This corresponds to the situation with normal paraxial machining, e.g., with G91Z ...

If the motion is programmed in the form MOVТ=AC (...), MOVT functions absolutely. In this case a plane is defined, which runs through the current zero point, and whose surface normal vector is parallel to the tool orientation. MOVT then gives the position relative to this plane:

Figure 18-48 Definition of the position for absolute programming of a motion in tool direction

The reference to this auxiliary plane serves only to calculate the end position. Active frames are not affected by this internal calculation.
Instead of MOVT= ... it is also possible to write MOVT=IC( ...) if it is to be plainly visible that MOVT is to function incrementally. There is no functional difference between the two forms.

Supplementary conditions

The following supplementary conditions apply to programming with MOVT:

- It is independent of the existence of a toolholder with orientation capability. The direction of the motion is dependent on the active plane. It runs in the direction of the vertical axes, i.e., with G17 in Z direction, with G18 in Y direction and with G19 in X direction. This applies both where no toolholder with orientation capability is active and for the case of a toolholder with orientation capability without rotary tool or with a rotary tool in its basic setting.

- MOVT acts similarly for active orientation transformation (345axis transformation).

- If in a block with MOVT the tool orientation is changed simultaneously (e.g., active 5axis transformation by means of simultaneous interpolation of the rotary axes), the orientation at the start of the block is decisive for the direction of movement of MOVT. The path of the tool tip (TCP - Tool Center Point) is not affected by the change in orientation.

- Linear or spline interpolation (G0, G1, ASPLINE, BSPLINE, CSPLINE) must be active. Otherwise, an alarm is produced. If a spline interpolation is active, the resultant path is generally not a straight line, since the end point determined by MOVT is treated as if it had been programmed explicitly with X, Y, Z.

- A block with MOVT must not contain any programming of geometry axes (alarm 14157).

18.9 Basic tool orientation

Application

Normally, the orientation assigned to the tool itself depends exclusively on the active machining plane. For example, the tool orientation is parallel to Z with G17, parallel to Y with G18 and parallel to X with G19.

Different tool orientations can only be programmed by activating a 5axis transformation. The following system variables have been introduced in order to assign a separate orientation to each tool cutting edge:

<table>
<thead>
<tr>
<th>System variable</th>
<th>Description of tool orientation</th>
<th>Format</th>
<th>Preassignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_DPV[t, d]</td>
<td>Tool cutting edge orientation</td>
<td>INT</td>
<td>0</td>
</tr>
<tr>
<td>$TC_DPV3[t, d]</td>
<td>L1 component of tool orientation</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>$TC_DPV4[t, d]</td>
<td>L2 component of tool orientation</td>
<td>REAL</td>
<td>0</td>
</tr>
<tr>
<td>$TC_DPV5[t, d]</td>
<td>L3 component of tool orientation</td>
<td>REAL</td>
<td>0</td>
</tr>
</tbody>
</table>

Indexing: Same as tool system variable $TC_DPx[t, d]

- t: T number of cutting edge
- d: D number of cutting edge
Identifiers $TC_{DPV3}$ to $TC_{DPV5}$ are analogous to identifiers $TC_{DP3}$ to $TC_{DP5}$ of the tool length components.

**MD18114**

The system variables for describing the tool orientation are only available if machine data is not equal to zero:

MD18114 $MN_{MM\_ENABLE\_TOOL\_ORIENT}$ (assign orientation to tool cutting)

<table>
<thead>
<tr>
<th>MD18114 $MN_{MM_ENABLE_TOOL_ORIENT}$</th>
<th>Value = 1</th>
<th>Only system variable $TC_DPV[t, d]$ is available.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value = 2</td>
<td>All four system variables are available.</td>
</tr>
</tbody>
</table>

**Define direction vector**

If all four system variables contain 0, the orientation is defined only by the active plane (as before).

If system variable $TC\_DPV[t, d]$ is equal to zero, the other three parameters - if available - define a direction vector. The amount of the vector is insignificant.

**Example:**

\[
\begin{align*}
&T_{C\_DPV[1, 1]} = 0 \\
&T_{C\_DPV3[1, 1]} = 1.0 \\
&T_{C\_DPV4[1, 1]} = 0.0 \\
&T_{C\_DPV5[1, 1]} = 1.0
\end{align*}
\]

In this example, the basic orientation points in the direction of the bisector in the L1L3 plane, i.e., the basic orientation in the bisector for a milling tool and active plane G17 lies in the Z/X plane.

**Basic orientation of tools**

<table>
<thead>
<tr>
<th>Basic orientation of:</th>
<th>With :</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turning and grinding tools</td>
<td>G18</td>
</tr>
<tr>
<td>Milling tools</td>
<td>G17</td>
</tr>
</tbody>
</table>

The active tool orientation is unchanged in these cases and is equivalent to the original settings in $TC\_DPVx[t, d]$.

The basic orientation is always the direction perpendicular to the plane in which tool radius compensation is performed. With turning tools, in particular, the tool orientation generally coincides with the longitudinal tool axis.

The setting data specified below are effective only if the basic orientation of the tool is defined by an entry in at least one of the system variables $TC\_DPVx[t, d]$.  

Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2

1485
They have no effect if the tool orientation is only determined by the plane selection G17 - G19 and is compatible with previous behavior.

The plane of the basic orientation for a cutting edge is treated either like a milling tool or like a turning tool, irrespective of the entry in $TC_DP1, if the following setting data is not equal to zero:

SD42950 $SC_TOOL_LENGTH_TYPE (allocation of the tool length components independent of tool type)

**Plane change**

A change of plane causes a change in orientation.

The following rotations are initiated:

<table>
<thead>
<tr>
<th>When changing from:</th>
<th>Rotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>G17 ⇒ G18:</td>
<td>Rotation through -90 degrees about the Z axis followed by rotation through -90 degrees about the X axis</td>
</tr>
<tr>
<td>G18 ⇒ G19:</td>
<td>Rotation through 90 degrees about the X axis followed by rotation through 90 degrees about the Z axis</td>
</tr>
<tr>
<td>G19 ⇒ G17:</td>
<td></td>
</tr>
</tbody>
</table>

These rotations are the same as those that have to be performed in order to interchange the components of the tool length vector on a change of plane.

The basic orientation is also rotated when an adapter transformation is active.

If the following setting data is not equal to zero, the tool orientation is not rotated on a change of plane:

SD42940 $SC_TOOL_LENGTH_CONST (change of tool length components on change of planes).

**Tool length components**

The components of the tool orientation are treated the same as the components of the tool length, with respect to setting data:

SD42910 $SC_MIRROR_TOOL_LENGTH (Sign change tool wear when mirroring).

SD42950 $SC_TOOL_LENGTH_TYPE (allocation of the tool length components independent of tool type)

Therefore the components are changed respectively and assigned to the geometry axis.
System variable $TC_DPV[t, d]$

The purpose of system variable $TC_DPV[t, d]$ is to allow the simple specification of certain basic orientations (parallel to coordinate axes) that are required frequently. The permissible values are shown in the table below. The values in the first and second/third columns are equivalent.

<table>
<thead>
<tr>
<th>$TC_DPV[t, d]$</th>
<th>Milling tools *</th>
<th>Turning tools *</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ 0 or &gt; 6</td>
<td>($TC_DPV5[t, d], $TC_DPV4[t, d], $TC_DPV3[t, d]$,) **</td>
<td>($TC_DPV3[t, d], $TC_DPV5[t, d], $TC_DPV4[t, d],$TC_DPV5[t, d],) **</td>
</tr>
<tr>
<td>1</td>
<td>(0, 0, V)</td>
<td>(0, V, 0)</td>
</tr>
<tr>
<td>2</td>
<td>(0, V, 0)</td>
<td>(0, 0, V)</td>
</tr>
<tr>
<td>3</td>
<td>(V, 0, 0)</td>
<td>(V, 0, 0)</td>
</tr>
<tr>
<td>4</td>
<td>(0, 0, -V)</td>
<td>(0, -V, 0)</td>
</tr>
<tr>
<td>5</td>
<td>(0, -V, 0)</td>
<td>(0, -V, 0)</td>
</tr>
<tr>
<td>6</td>
<td>(-V, 0, 0)</td>
<td>(-V, 0, 0)</td>
</tr>
</tbody>
</table>

* Turning tools in this context are any tools whose tool type ($TC_DP1[t, d]$) is between 400 and 599. All other tool types refer to milling tools.

** If all three values $TC_DPV3[t, d], $TC_DPV4[t, d], $TC_DPV5[t, d]$ are equal to zero in this case, the tool orientation is determined by the active machining plane (default).

V Stands for a positive value in the corresponding system variables.

Example:

For milling tools:

$TC_DPV[t, d] = 2$ is equal to:

$TC_DPV3[t, d] = 0, $TC_DPV4[t, d] = 0, $TC_DPV5[t, d] = V.$

Supplementary conditions

If the "Scratch" function is used in the **RESET** state, the following must be noted with respect to the initial setting:

- The wear components are evaluated depending on the initial settings of the G-code groups TOWSTD, TOWNCS and TOWNCS.

- If a value other than the initial setting is needed to ensure correct calculation, scratching may be performed only in the **STOP** state.

**Note**

"Special handling of tool compensations" pays particular attention to tool compensations with evaluation of sign for tool length with wear and temperature fluctuations.

The following are taken into account:

- Tool type
- Transformations for tool components
- Assignment of tool length components to geometry axes independently of tool type

Basic Functions

Function Manual, 01/2015, 6FC5397-0BP40-5BA2
18.10 Special handling of tool compensations

18.10.1 Relevant setting data

**SD42900-42960**

Setting data SD42900 - SD42940 can be used to make the following settings with reference to tool compensation:

- Sign of the tool length
- Sign of the wear
- Behavior of the wear components when mirroring geometry axes
- Behavior of the wear components when changing the machining plane via setting data
- Allocation of the tool length components independent of actual tool type
- Transformation of wear components into a suitable coordinate system for controlling the effective tool length

**Note**

In the following description, the wear includes the total values of the following components:

- Wear values: $TC_{DP12}$ to $TC_{DP20}$
- Sum offset, consisting of:
  - Wear values: $SCPX3$ to $SCPX11$
  - Setup values: $ECPX3$ to $ECPX11$

You will find detailed information about sum and tool offsets in:

**References:**
- Function Manual Tool Management
- Programming Manual. Fundamentals; Tool Offsets

**Required setting data**

- SD42900 $SC\_MIRROR\_TOOL\_LENGTH$ (mirroring of tool length components and components of the tool base dimension)
- SD42910 $SC\_MIRROR\_TOOL\_WEAR$ (mirroring of wear values of tool length components)
- SD42920 $SC\_WEAR\_SIGN\_CUTPOS$ (sign evaluation of the wear components)
- SD42930 $SC\_WEAR\_SIGN$ (inverts the sign of the wear dimensions)
- SD42940 $SC\_TOOL\_LENGTH\_CONST$ (allocation of the tool length components to the geometry axes)
- SD42950 $SC\_TOOL\_LENGTH\_TYPE$ (allocation of the tool length components independent of tool type)
18.10 Special handling of tool compensations

18.10.2 Mirror tool lengths (SD42900 $SC_MIRROR_TOOL_LENGTH)

Activation

Tool length mirroring is activated via the setting data:

SD42900 $SC_MIRROR_TOOL_LENGTH <> 0 (TRUE) (Sign change tool length when mirroring)

Function

The following components are mirrored by inverting the sign:

- Tool lengths: $TC_DP3, $TC_DP4, $TC_DP5
- Tool base dimensions: $TC_DP21, $TC_DP22, $TC_DP23

Mirroring is performed for all tool base dimensions whose associated axes are mirrored. Wear values are not mirrored.

Mirror wear values

The following setting data should be set in order to mirror the wear values:

SD42910 $SC_MIRROR_TOOL_WEAR <> 0 (Sign change tool wear when mirroring)

Inverting the sign mirrors the wear values of the tool length components whose associated axes are mirrored.

Figure 18-49 Application example: Double-spindle turning machine
18.10.3 Mirror wear lengths (SD42920 $SC\_WEAR\_SIGN\_CUTPOS)

Activation

Wear length mirroring is activated by:

SD42920 $SC\_WEAR\_SIGN\_CUTPOS <> 0 (TRUE) (Sign of wear for tools with cutting edge systems)

Function

<table>
<thead>
<tr>
<th>Length of cutting edge</th>
<th>Length 1</th>
<th>Length 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>2</td>
<td>---</td>
<td>Inverted</td>
</tr>
<tr>
<td>3</td>
<td>Inverted</td>
<td>Inverted</td>
</tr>
<tr>
<td>4</td>
<td>Inverted</td>
<td>---</td>
</tr>
<tr>
<td>5</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>6</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>7</td>
<td>---</td>
<td>Inverted</td>
</tr>
<tr>
<td>8</td>
<td>Inverted</td>
<td>---</td>
</tr>
<tr>
<td>9</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

In the case of tool types without a relevant cutting edge position, the wear length is not mirrored.

Note

The mirroring (sign inversion) in one or more components can cancel itself through a simultaneous activation of the functions:

Tool length-mirroring (SD42900 <> 0)
And:

Tool length-mirroring (SD42920 <> 0)

SD42930 $SC\_WEAR\_SIGN

Setting data not equal to zero:

Inverts the sign of all wear dimensions. This affects both the tool length and other variables such as tool radius, rounding radius, etc.

Entering a positive wear dimension makes the tool "shorter" and "thinner".

Activation of modified setting data

When the setting data described above are modified, the tool components are not reevaluated until the next time a tool edge is selected. If a tool is already active and the data of this tool are to be reevaluated, the tool must be selected again.
18.10 Special handling of tool compensations

Example:

```
N10 $SC_WEAR_SIGN = 0 ; No sign inversion of the wear values
N20 $TC_DP1[1,1] = 120 ; End mill
N30 $TC_DP6[1,1] = 100 ; Tool radius 100 mm
N40 $TC_DP15[1,1] = 1 ; Wear dimension of tool radius 1 mm, resulting tool radius 101 mm

N100 T1 D1 G41 X150 Y20
....
N150 G40 X300 N10
....
N200 $SC_WEAR_SIGN = 1 ; Sign inversion for all wear values; the new radius of 99 mm is activated on a new selection (D1). Without D1, the radius would continue to be 101 mm.

N300 D1 G41 X350 Y-20
N310 ....
```

The same applies in the event that the resulting tool length is modified due to a change in the mirroring status of an axis. The tool must be selected again after the mirror command, in order to activate the modified tool-length components.

18.10.4 Tool length and plane change (SD42940 $SC_TOOL_LENGTH_CONST)

Plane change

The assignment of tool length components (length, wear and tool base dimension) to geometry axes does not change when the machining plane is changed (G17–G19).

Assignment of tools

The assignment of tool length components to geometry axes for turning and grinding tools (tool types 400 to 599) is generated from the value of the following setting data in accordance with the following table:

SD42940 $SC_TOOL_LENGTH_CONST (change of tool length components on change of planes).

<table>
<thead>
<tr>
<th>Layer</th>
<th>Length 1</th>
<th>Length 2</th>
<th>Length 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Y</td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>*)</td>
<td>X</td>
<td>Z</td>
<td>Y</td>
</tr>
<tr>
<td>19</td>
<td>Z</td>
<td>Y</td>
<td>X</td>
</tr>
<tr>
<td>-17</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>-18</td>
<td>Z</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>-19</td>
<td>Y</td>
<td>Z</td>
<td>X</td>
</tr>
</tbody>
</table>
The following table shows the assignment of tool length components to geometry axes for all other tools (tool types < 400 or > 599):

<table>
<thead>
<tr>
<th>Layer</th>
<th>Length 1</th>
<th>Length 2</th>
<th>Length 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>*)</td>
<td>Z</td>
<td>Y</td>
<td>X</td>
</tr>
<tr>
<td>18</td>
<td>Y</td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>19</td>
<td>X</td>
<td>Z</td>
<td>Y</td>
</tr>
<tr>
<td>-17</td>
<td>Z</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>-18</td>
<td>Y</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>-19</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>

*) Each value not equal to 0 and not equal to one of the six listed values is evaluated as value 17.

Note

For representation in tables, it is assumed that geometry axes 1 to 3 are named X, Y, Z. The axis order and not the axis name determines the assignment between a compensation and an axis.

Three tool length components can be arranged on the six different types above.

18.10.5 Tool type (SD42950 $SC_TOOL_LENGTH_TYPE)

Characteristics

Definition of the assignment between tool length components (length, wear and tool base dimension) and geometry axes independent of tool type.

Setting data not equal to zero: (the default definition is applied)

A distinction is made between turning and grinding tools (tool types 400 to 599) and other tools (milling tools).

The value range is from 0 to 2. Any other value is interpreted as 0.

The assignment of tool length components is always independent of the actual tool type.

- With value = 1: Always as for milling tools
- With value = 2: Always as for turning tools

Toolholder with orientation capability

Setting data SD42900 - SD42950
Setting data SD42900 - SD42950 have no effect on the components of an active toolholder with orientation capability. The calculation with a toolholder with orientation capability always allows for a tool with its total resulting length (tool length + wear + tool base dimension). The calculation of the resulting total length allows for all modifications caused by the setting data.

**Note**

When toolholders with orientation capability are used, it is common to define all tools for a non-mirrored basic system, even those, which are only used for mirrored machining. When machining with mirrored axes, the toolholder is then rotated such that the actual position of the tool is described correctly. All tool-length components then automatically act in the correct direction, dispensing with the need for control of individual component evaluation via setting data, depending on the mirroring status of individual axes.

The use of toolholders with orientation capability is also practical if the physical characteristics of the machine type prevents tools, which are permanently installed with different orientations, from being rotated. Tool dimensioning can then be performed uniformly in a basic orientation, where the dimensions relevant for machining are calculated according to the rotations of a virtual toolholder.

### 18.10.6 Tool lengths in the WCS, allowing for the orientation

**Change tool or working plane**

The values displayed for the tool correspond to the expansion in the WCS. If a toolholder with an inclined clamping position is to be used, you should make sure that the transformation used supports the toolholder. If this is not the case, incorrect tool dimensions will be displayed. When changing the working plane from G17 to G18 or G19, you should ensure that the transformation can also be used for these working planes. If the transformation is only available for G17 machining, the dimensions continue to be displayed for a tool in the Z direction after the plane change.

When transformation is deactivated, the basic tool is displayed in the x, y or z direction, according to the working plane. Allowance is made for a programmed toolholder. These tool dimensions are not altered when traversing without a transformation.

### 18.10.7 Tool length offsets in tool direction

**Temperature compensation in real time**

On 5-axis machines with a moving tool, temperature fluctuations can occur in the machining heads. These can result directly in expansion fluctuations which are transmitted to the tool spindle in the form of linear expansion. A typical case on 5-axis heads, for example, is thermal expansion in the direction of the longitudinal spindle axis.

It is possible to compensate this thermal expansion even when the tool is orientated by assigning the temperature compensation values to the tool rather than to the machine axes.
In this way, linear expansion fluctuations can be compensated even when the tool orientation changes.

Using the orientation transformation whose direction is determined by the current tool orientation, it is possible to overlay motions in real time and rotate them simultaneously. At the same time, the compensation values are adjusted continuously in the tool coordinate system.

The temperature compensation only becomes effective if the axis to be compensated is really referenced.

**Activation**

The temperature compensation in the tool direction is activated by setting the following machine data to a value **not equal to zero**.

MD20390 $MC_TOOL_TEMP_COMP_ON (activation of temperature compensation for tool length)

In addition, bit 2 must be set for each affected channel axis in the machine data:

MD32750 $MA_TEMP_COMP_TYPE [<axis index>] (temperature compensation type)

This can be more than three axes in cases where more than three channel axes in succession can be temporarily assigned to geometry axes as a result of geometry axis replacement of transformation switchover. If this bit is not set for a particular channel axis, the compensation value cannot be applied in the axis. This does not have any effect on other axes. In this case, an alarm is not output.

**Applicability**

The temperature compensation in the tool direction is only effective with generic 5-axis transformations with:

- Transformation type 24
  Two axes rotate the tool
- Transformation type 56
  One axis rotates the tool, the other axis rotates the workpiece without temperature compensation

In generic 5-axis transformation with:

- Transformation type 40
  The tool orientation is constant with a rotary workpiece, which means that the motion of the rotary axes on the machine does not affect the temperature compensation direction.

Temperature compensation in the tool direction also works in conjunction with orientation transformations (not generic 5-axis transformations) with:

- Transformation type 64 to 69
  Rotating linear axis

**Note**

Temperature compensation can be activated with all other types of transformation. It is not affected by a change in tool orientation. The axes move as if no orientation transformation with temperature compensation were active.
**Limit values**

The compensation values are restricted to the maximum values by the machine data:
MD20392 $MC_TOOL_TEMP_COMP_LIMIT[0...2] (maximum temperature compensation for tool length)

The limit value default setting is 1 mm. If a temperature compensation value higher than this limit is specified, it will be limited without an alarm.

**SD42960**

The three temperature compensation values together form a compensation vector and are contained in setting data:
SD42960 $SC_TOOL_TEMP_COMP[0...2] (temperature compensation with reference to tools)

The setting data is user-defined, e.g. using synchronized actions or from the PLC. The compensation values can, therefore, also be used for other compensation purposes.

In the initial state or when orientation transformation is deactivated, all three compensation values apply in the direction of the three geometry axes (in the typical order X, Y, Z). The assignment of components to geometry axes is independent of the tool type (turning, milling or grinding tools) and the selected machining plane G17 to G19. Changes to the setting data values take effect immediately.

**Toolholder with orientation capability**

If a toolholder with orientation capability is active, the temperature compensation vector is rotated simultaneously to any change in orientation. This applies independently of any active orientation transformation.

If a toolholder with orientation capability is active in conjunction with a generic 5-axis transformation or a transformation with rotating linear axis, the temperature compensation vector is subjected to both rotations.

---

**Note**

While transformations with rotating linear axes take changes in the tool vector (length) into account, they **ignore** its change in orientation, which can be effected by a toolholder with orientation capability.

Temperature compensation values immediately follow any applied change in orientation. This applies in particular when an orientation transformation is activated or deactivated.

The same is true when the assignment between geometry axes and channel axes is changed. The temperature compensation value for an axis is reduced to zero (interpolatively), for example, when it ceases to be a geometry axis after a transformation change. Conversely, any temperature compensation value for an axis which changes over to geometry axis status is applied immediately.
Examples

Temperature compensation in tool direction

Example of a 5-axis machine with rotating tool on which the tool can be rotated around the C and B axes.

In its initial state, the tool is parallel to the Z axis. If the B axis is rotated through 90 degrees, the tool points in the X direction.

Therefore, a temperature compensation value in the following setting data is also effective in the direction of the machine X axis if transformation is active:

SD42960 $SC_TOOL_TEMP_COMP[2]$ (temperature compensation with reference to tools)

If the transformation is deactivated with the tool in this direction, the tool orientation is, by definition, parallel again to the Z axis and thus different to its actual orientation. The temperature offset in the X axis direction is therefore reduced to zero and reapplied simultaneously in the Z direction.

Example of a 5-axis machine with rotating tool (transformation type 24). The relevant machine data is listed below:

- The first rotary axis rotates around Z, C-axis
- The second rotary axis rotates around Y, B-axis

The essential machine data is shown in the table below:

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Value</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD20390 $MC_TOOL_TEMP_COMP_ON</td>
<td>= TRUE</td>
<td>Temperature compensation active</td>
</tr>
<tr>
<td>MD32750 $MA_TEMP_COMP_TYPE[ AX1 ]</td>
<td>= 4</td>
<td>Compensation in tool direction</td>
</tr>
<tr>
<td>MD32750 $MA_TEMP_COMP_TYPE[ AX2 ]</td>
<td>= 4</td>
<td>Compensation in tool direction</td>
</tr>
<tr>
<td>MD32750 $MA_TEMP_COMP_TYPE[ AX3 ]</td>
<td>= 4</td>
<td>Compensation in tool direction</td>
</tr>
<tr>
<td>MD24100 $MC_TRAFO_TYPE_1</td>
<td>= 24</td>
<td>Assignment of transformation type 24</td>
</tr>
<tr>
<td>MD24110 $MC_TRAFO_AXES_IN_1[0]</td>
<td>= 1</td>
<td>First axis of the transformation</td>
</tr>
<tr>
<td>MD24110 $MC_TRAFO_AXES_IN_1[1]</td>
<td>= 2</td>
<td>Second axis of the transformation</td>
</tr>
<tr>
<td>MD24110 $MC_TRAFO_AXES_IN_1[2]</td>
<td>= 3</td>
<td>Third axis of the transformation</td>
</tr>
<tr>
<td>MD24110 $MC_TRAFO_AXES_IN_1[3]</td>
<td>= 5</td>
<td>Fifth axis of the transformation</td>
</tr>
<tr>
<td>MD24110 $MC_TRAFO_AXES_IN_1[4]</td>
<td>= 4</td>
<td>Fourth axis of the transformation</td>
</tr>
<tr>
<td>MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]</td>
<td>= 1</td>
<td>Geometry axis for channel axis 1</td>
</tr>
<tr>
<td>MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]</td>
<td>= 2</td>
<td>Geometry axis for channel axis 2</td>
</tr>
<tr>
<td>MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]</td>
<td>= 3</td>
<td>Geometry axis for channel axis 3</td>
</tr>
<tr>
<td>MD24570 $MC_TRAFO5_AXIS1_1[0]</td>
<td>= 0.0</td>
<td>Direction</td>
</tr>
<tr>
<td>MD24570 $MC_TRAFO5_AXIS1_1[1]</td>
<td>= 0.0</td>
<td>Direction</td>
</tr>
</tbody>
</table>
### Machine data

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Value</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD24570 $MC.TRAFO5_AXIS1_1[2]</td>
<td>= 1.0</td>
<td>First rotary axis is parallel to Z</td>
</tr>
<tr>
<td>MD24572 $MC.TRAFO5_AXIS1_2[0]</td>
<td>= 0.0</td>
<td>Direction</td>
</tr>
<tr>
<td>MD24572 $MC.TRAFO5_AXIS1_2[1]</td>
<td>= 1.0</td>
<td>Second rotary axis is parallel to Y</td>
</tr>
<tr>
<td>MD24572 $MC.TRAFO5_AXIS1_2[2]</td>
<td>= 0.0</td>
<td></td>
</tr>
<tr>
<td>MD25574 $MC.TRAFO5_BASE_ORIENT_1[0]</td>
<td>= 0.0</td>
<td></td>
</tr>
<tr>
<td>MD25574 $MC.TRAFO5_BASE_ORIENT_1[1]</td>
<td>= 0.0</td>
<td>Basic tool orientation</td>
</tr>
<tr>
<td>MD25574 $MC.TRAFO5_BASE_ORIENT_1[2]</td>
<td>= 1.0</td>
<td>In Z direction</td>
</tr>
</tbody>
</table>

### Temperature compensation values in the NC program

The compensation values assigned to axes X and Z are not zero and are applied for temperature compensation with respect to tool length. The machine axis positions reached in each case are specified as comments in the program lines.

<table>
<thead>
<tr>
<th>Program code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SC_TOOL_TEMP_COMP[0] = -0.3</td>
<td>; Compensation value in X</td>
</tr>
<tr>
<td>$SC_TOOL_TEMP_COMP[1] = 0.0</td>
<td>;</td>
</tr>
<tr>
<td>$SC_TOOL_TEMP_COMP[2] = -1.0</td>
<td>; Compensation value in Z</td>
</tr>
<tr>
<td>N10 G74 X0 Y0 Z0 A0 B0</td>
<td>; X Y Z</td>
</tr>
<tr>
<td>N20 X20 Y20 Z20 F10000</td>
<td>; 20.30 20.00 21.00</td>
</tr>
<tr>
<td>N30 TRAORI()</td>
<td>; 20.30 20.00 21.00</td>
</tr>
<tr>
<td>N40 X10 Y10 Z10 B90</td>
<td>; 11.00 10.00 9.70</td>
</tr>
<tr>
<td>N50 TRAFGOF</td>
<td>; 10.30 10.00 11.00</td>
</tr>
<tr>
<td>N60 X0 Y0 Z0 B0 C0</td>
<td>; 0.30 0.00 1.00</td>
</tr>
<tr>
<td>N70 M30</td>
<td></td>
</tr>
</tbody>
</table>

With the exception of block N40, temperature compensation always acts in the original directions, as the tool is pointing in the basic orientation direction. This applies particularly in block N50. The tool is actually still pointing in the direction of the X axis because the B axis is still at 90 degrees. However, because the transformation is already deactivated, the applied orientation is parallel to the Z axis again.

### Machine data

<table>
<thead>
<tr>
<th>Machine data</th>
<th>Value</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD20390 $MC_TOOL_TEMP_COMP_ON</td>
<td>= TRUE</td>
<td>Temperature compensation active</td>
</tr>
<tr>
<td>MD32750 $MA_TEMP_COMP_TYPE[ AX1 ]</td>
<td>= 4</td>
<td>Compensation in tool direction</td>
</tr>
<tr>
<td>MD32750 $MA_TEMP_COMP_TYPE[ AX2 ]</td>
<td>= 4</td>
<td>Compensation in tool direction</td>
</tr>
<tr>
<td>MD32750 $MA_TEMP_COMP_TYPE[ AX3 ]</td>
<td>= 4</td>
<td>Compensation in tool direction</td>
</tr>
</tbody>
</table>

### Additional references

For more details on “Temperature compensation” see:

**References:**
Function Manual Extended Functions; Compensations (K3)
For information on "Generic 5-axis transformations" see:

References:
Function Manual, Special Functions; 3- to 5-Axis Transformation (F2)

18.11 Sum offsets and setup offsets

18.11.1 General

Sum offsets

Sum offsets can be treated as programmable process compensations during machining and are composed of all the error sizes (including the wear), which cause the workpiece to deviate from the specified dimensions.

Sum offsets are a generalized type of wear. They are part of the cutting edge data. The parameters of the sum offset refer to the geometrical data of a cutting edge.

The compensation data of a sum offset are addressed by a DL number (DL: location-dependent; compensations with reference to the location of use).

In contrast, the wear values of a D number describe the physical wear of the cutting edge, i.e., in special situations, the sum offset can match the wear of the cutting edge.

Sum offsets are intended for general use, i.e., with active or inactive tool management or with the flat D number function.

Machine data are used to classify the sum offsets into:

- Sum offset fine
- Sum offset coarse (setup offset)

Setup offset

The setup offset is the compensation to be entered by the setup engineer before machining. These values are stored separately in the NCK. The operator subsequently only has access to the "sum offset fine" via HMI.

The "sum offset fine" and "sum offset coarse" are added internally in the NCK. This value is referred to below as the sum offset.

Note

The function is enabled via the machine data setting:

MD18080 $MN_MM_TOOL_MANAGEMENT_MASK, Bit 8=1 (Gradual memory reservation for tool management).

If kinematic transformations (e.g., 5axis transformations) are active, the tool length is calculated first after allowing for the various wear components. The total tool length is then
used in the transformation. Unlike the case of a toolholder with orientation capability, the wear values are thus always included in the transformation irrespective of the G code of group 56.

18.11.2 Description of function

Sum offsets

Several sum offsets (DL numbers) can be defined per D number. This allows you to determine, for example, \texttt{workpiecelocationdependent} compensation values and assign them to a cutting edge. Sum offsets have the same effect as wear, i.e., they are added to the compensation values of the D number. The data are permanently assigned to a D number.

Attitudes

You can define the following settings in machine data:

- Activate sum offset
- Define maximum quantity of DL data sets to be created in NCK memory
- Define maximum quantity of DL numbers to be assigned to a D number
- Define whether the sum offsets (fine/coarse) are to be saved during data backup
- Define the sum offset to be activated, if:
  - A new cutting edge compensation is activated
  - An operator panel front \texttt{RESET} is performed
  - An operator panel front \texttt{START} is performed
  - The end of the program has been reached

The name is oriented to the logic of the corresponding machine data for tools and cutting edges. The "setup offset" and "sum offset fine" can be read and written via system variables and corresponding OPI services.

Note

When tool management is active, a machine data can be used to define whether the sum offset of a tool activated during a programmed tool change remains unchanged or is set to zero.

Summary of compensation parameters $TC_{DPx}$

The following general system variables were previously defined for describing a cutting edge:

| $TC_{DP1}$ | Tool type |
| $TC_{DP2}$ | Length of cutting edge |
Parameters for geometry and wear

Tool geometry compensations are assigned to system variables $TC_DP3 to $TC_DP11. System variables $TC_DP12 to $TC_DP20 allow you to name a wear for each of these parameters.

<table>
<thead>
<tr>
<th>Geometry</th>
<th>Wear</th>
<th>Length compensations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_DP3</td>
<td>$TC_DP12</td>
<td>Length 1</td>
</tr>
<tr>
<td>$TC_DP4</td>
<td>$TC_DP13</td>
<td>Length 2</td>
</tr>
<tr>
<td>$TC_DP5</td>
<td>$TC_DP14</td>
<td>Length 3</td>
</tr>
<tr>
<td>Geometry</td>
<td>Wear</td>
<td>Radius compensation</td>
</tr>
<tr>
<td>$TC_DP6</td>
<td>$TC_DP15</td>
<td>Radius</td>
</tr>
<tr>
<td>$TC_DP7</td>
<td>$TC_DP16</td>
<td>Corner radius (tool type 700; slotting saw)</td>
</tr>
<tr>
<td>Geometry</td>
<td>Wear</td>
<td>Further compensations</td>
</tr>
<tr>
<td>$TC_DP8</td>
<td>$TC_DP17</td>
<td>Length 4 (tool type 700; slotting saw)</td>
</tr>
<tr>
<td>$TC_DP9</td>
<td>$TC_DP18</td>
<td>Length 5</td>
</tr>
<tr>
<td>$TC_DP10</td>
<td>$TC_DP19</td>
<td>Angle 1 (angle between face of tool and torus surface)</td>
</tr>
<tr>
<td>$TC_DP11</td>
<td>$TC_DP20</td>
<td>Angle 2 (angle between tool longitudinal axis and upper end of torus surface)</td>
</tr>
</tbody>
</table>

Tool base dimension/adapter dimension

| $TC_DP21 | Adapter length 1 |
| $TC_DP22 | Adapter length 2 |
| $TC_DP23 | Adapter length 3 |

Technology

<table>
<thead>
<tr>
<th>System variable</th>
<th>Clearance angle</th>
</tr>
</thead>
</table>
| $TC_DP24        | - The clearance angle is stored here for ManualTurn; tool type 5xx. Same significance as in standard cycles for turning tools.  
    - The tip angle of the drill is stored here for ShopMill; tool type 2xx.  
    - Used in standard cycles for turning tools; tool type 5xx. This is the angle at the secondary cutting edge for these tools.  |
| $TC_DP25        | - The value for the cutting rate is stored here for ManualTurn.  
    - A bitcoded value for various states of tool types 1xx and 2xx is stored here for ShopMill.  |

Parameters of the sum and setup offsets ($TC_SCPxy, $TC_ECPxy)

The numbering of the parameters is oriented to the numbering of system variables $TC_DP3 to $TC_DP11.
The effect of the parameters is similar to the wear (additive to the tool geometry). Up to six sum/setup parameters can be defined per cutting edge parameter.

<table>
<thead>
<tr>
<th>Tool geometry parameter, to which the compensation is added.</th>
<th>Sum/setup parameters, length compensations</th>
<th>Tool wear parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_{DP3}$</td>
<td>Length 1</td>
<td>$TC_{DP12}$</td>
</tr>
<tr>
<td></td>
<td>$TC_{SCP13}, TC_{SCP23}, TC_{SCP33},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$TC_{SCP43}, TC_{SCP53}, TC_{SCP63}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$TC_{ECP13}, TC_{ECP23}, TC_{ECP33},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$TC_{ECP43}, TC_{ECP53}, TC_{ECP63}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The numbers in bold, 1, 2, ... 6, designate the parameters of a maximum of six (location-dependent or similar) compensations that can be programmed with DL = 1 to 6 for the parameter specified in column one.</td>
<td></td>
</tr>
<tr>
<td>$TC_{DP4}$</td>
<td>Length 2</td>
<td>$TC_{DP13}$</td>
</tr>
<tr>
<td></td>
<td>$TC_{SCP14}, TC_{SCP24}, TC_{SCP34},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$TC_{SCP44}, TC_{SCP54}, TC_{SCP64}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$TC_{ECP14}, TC_{ECP24}, TC_{ECP34},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$TC_{ECP44}, TC_{ECP54}, TC_{ECP64}</td>
<td></td>
</tr>
<tr>
<td>$TC_{DP5}$</td>
<td>Length 3</td>
<td>$TC_{DP14}$</td>
</tr>
<tr>
<td></td>
<td>etc.</td>
<td></td>
</tr>
<tr>
<td>$TC_{DP6}$</td>
<td>Radius</td>
<td>$TC_{DP15}$</td>
</tr>
<tr>
<td>$TC_{DP7}$</td>
<td>Corner radius</td>
<td>$TC_{DP16}$</td>
</tr>
<tr>
<td>Further compensations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_{DP8}$</td>
<td>Length 4</td>
<td>$TC_{DP17}$</td>
</tr>
<tr>
<td>$TC_{DP9}$</td>
<td>Length 5</td>
<td>$TC_{DP18}$</td>
</tr>
<tr>
<td>$TC_{DP10}$</td>
<td>Angle 1,</td>
<td>$TC_{DP19}$</td>
</tr>
<tr>
<td></td>
<td>etc.</td>
<td></td>
</tr>
<tr>
<td>$TC_{DP11}$</td>
<td>Angle 2</td>
<td>$TC_{DP20}$</td>
</tr>
<tr>
<td></td>
<td>$TC_{SCP21}, TC_{SCP31}, TC_{SCP41},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$TC_{SCP51}, TC_{SCP61}, TC_{SCP71}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$TC_{ECP21}, TC_{ECP31}, TC_{ECP41},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$TC_{ECP51}, TC_{ECP61}, TC_{ECP71}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The numbers in bold, 2, 3, ... 7, designate the parameters of a maximum of six (location-dependent or similar) compensations that can be programmed with DL = 1 to 6 for the parameter specified in column one.</td>
<td></td>
</tr>
</tbody>
</table>

**Supplementary conditions**

The maximum number of DL data sets of a cutting edge and the total number of sum offsets in the NCK are defined by machine data. The default value is zero, i.e., no sum offsets can be programmed.

Activate the "monitoring function" to monitor a tool for wear or for "sum offset".

The additional sum/setup data sets use additional buffered memory. 8 bytes are required per parameter.
A sum-offset data set requires: 8 bytes * 9 parameters = 72 bytes
A setup data set requires an equal amount of memory. A certain number of bytes is also required for internal administration data.

18.11.3 Activation

Function

The function must be activated via the machine data:

MD18108 $MN_MM_NUM_SUMCORR (sum offsets in TO area).

System variables $TC_ECPx and $TC_SCPx and setup and sum offsets ("fine") defined via the OPI interface can be activated in the part program.

This is done by programming the language command DL="number".

When a new D number is activated, either a new DL number is programmed, or the DL number defined via the following machine data becomes active:

MD20272 $MC_SUMCORR_DEFAULT (basic setting of the additive offset without a program)

DL programming

The sum offset is always programmed relative to the active D number with the command:

DL = "n"

The sum offset "n" is added to the wear of the active D number.

Note
If you use "setup offset" and "sum offset fine", both compensations are combined and added to the tool wear.

The sum offset is deselected with the command:

DL = 0

Note
DL0 is not allowed. If compensation is deselected (D0 and T0), the sum offset also becomes ineffective.

Programming a sum offset that does not exist triggers an alarm, similar to programming a D compensation that does not exist.

Thus, only the defined wear remains part of the compensation (defined in system variables $TC_DP12 to $TC_DP20).

Programming a sum offset when a D compensation is active (also applies to deselection) has the same effect on the path as programming a D command. An active radius compensation will, therefore, lose its reference to adjacent blocks.
Configuration

MD18112 $MN_MM_KIND_OF_SUMCORR, bit 4=0: (Properties of sum offset in the TO area)
default setting:

Only one set of sum offsets exists per DL number.

We refer in general to the sum offset.

This describes the data represented by $TC_SCPx.

Figure 18-50 MD18112 $MN_MM_KIND_OF_SUMCORR, bit 4 = 0

Tool T = t is active. With the data in the figure, the following is programmed:

D2 ; Cutting edge offsets, i.e., $TC_DP3 to $TC_DP11 + wear
    ($TC_DP12 to $TC_DP20) + adapter dimension
...
DL=1 ; Sum offset 1 is added to the previous D2 compensations, i.e.,
       $TC_SCP13 to $TC_SCP21.
...
DL=2 ; Sum offset 2 is added to the D2 compensation instead of sum
       offset 1, i.e., $TC_SCP23 to $TC_SCP31.
...
DL=0 ; Deselection of sum offset;
       only the data of D2 remain active.

MD18112 $MN_MM_KIND_OF_SUMCORR, bit 4=1: Setup offsets are available

The sum offset is now composed of the "sum offset fine" (represented by $TC_SCPx) and the
setup offset (represented by $TC_ECPx). Two data sets therefore exist for one DL number.
The sum offset is calculated by adding the corresponding components ($TC_ECPx +
$TC_SCPx).
Tool T = t is active. With the data in the figure, the following is programmed:

\[ \text{D2} \]
; Cutting edge compensations, i.e., $TC\_DP3$ to $TC\_DP11$ + wear ($TC\_DP12$ to $TC\_DP20$) + adapter dimension

\[ \ldots \]

\[ \text{DL=1} \]
; Sum offset 1 is added to the previous D2 compensations, i.e., $TC\_ECP13$ + $TC\_SCP13$ to $TC\_ECP21$ + $TC\_SCP21$.

\[ \ldots \]

\[ \text{DL=2} \]
; Sum offset 2 is added to the D2 compensation instead of sum offset 1; i.e., $TC\_ECP23$ + $TC\_SCP23$, ..., $TC\_ECP31$ + $TC\_SCP31$

\[ \ldots \]

\[ \text{DL=0} \]
; Deselection of sum offset. Only the data of D2 remain active.

Reading/writing in the part program

The individual sets of sum offset parameters are differentiated according to the number ranges of system variable $TC\_SCP$.

The significance of the individual variables is similar to geometry variables $TC\_DP3$ to $TC\_DP11$. Only length 1, length 2 and length 3 are enabled for the basic functionality (variables $TC\_SCP13$ to $TC\_SCP15$ for the first sum offset of the cutting edge).

\[ \text{R5} = \text{TC\_SCP13[ t, d ]} \]
; Sets the value of the R parameter to the value of the first component of sum offset 1 for cutting edge (d) on tool (t).

\[ \text{R6} = \text{TC\_SCP21[ t, d ]} \]
; Sets the value of the R parameter to the value of the last component of sum offset 1 for cutting edge (d) on tool (t).

\[ \text{R50} = \text{TC\_SCP23[ t, d ]} \]
; Sets the value of the R parameter to the value of the first component of sum offset 2 for cutting edge (d) on tool (t).
$TC_{SCP43}\[ t, d \] = 1.234 ; Sets the value of the first component of sum offset 4 for cutting edge (d) on tool (t) to the value 1.234.

The above statements also apply to the setup offsets (if the NCK is configured with this option), i.e.,

R5 = $TC_{ECP13}\[ t, d \] ; Sets the value of the R parameter to the value of the first component of setup offset 1 for cutting edge (d) on tool (t).

R6 = $TC_{ECP21}\[ t, d \] ; Sets the value of the R parameter to the value of the last component of setup offset 1 for cutting edge (d) on tool (t).

Etc.

When working with setup offsets, "sum offsets fine" are written with the $TC_{SCPx}$ system variables.

Creating a new sum offset

If the compensation data set (x) does not yet exist, it is created on the first write operation to one of its parameters (y).

$TC_{SCPxy}\[ t, d \] = r.r ; Parameter y of sum offset x is assigned the value "r.r.". The other parameters of x have a value of zero.

When working with setup offsets, "sum offsets fine" are written with the $TC_{SCPx}$ system variables.

Note

When working with setup offsets, the data set for the setup offset is created when a data set is created for "sum offset fine", if a data set did not already exist for [t, d].

Creating a new setup offset

If the compensation data set (x) does not yet exist, it is created on the first write operation to one of its parameters (y).

$TC_{ECPxy}\[ t, d \] = r.r ; The value "r.r" is assigned to the parameter y of setup offset x. The other parameters of x have the value zero.
Note
When working with setup offsets, the data set for the “sum offset fine” is created when a data set is created for setup offsets, if a data set did not already exist for [t, d].

DEDL - Delete sum offset

Sum offsets are generally only relevant when machining with a cutting edge at a certain time at a certain location of the workpiece. You can use the NC language command DELDL to delete sum offsets from cutting edges (in order to release memory).

```plaintext
status = DELDL( t, d ) ; Deletes all sum offsets for cutting edge d on tool t.
                     ; t, d are optional parameters.
```

If d is not specified, all sum offsets of all cutting edges of tool t are deleted.

If d and t are not specified, all sum offsets for the cutting edges on all tools of the TO unit are deleted (for the channel, in which the command is programmed).

When working with setup offsets, the DELDL command deletes both the setup offset and the “sum offsets fine” of the specified cutting edge(s).

Note
The memory used for the data sets is released after deletion.
The deleted sum offsets can subsequently no longer be activated or programmed.

Sum offsets and setup offsets on active tools cannot be deleted (similar to the deletion of D compensations or tool data).

The "status" return value indicates the result of the deletion command:

<table>
<thead>
<tr>
<th>status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Deletion was successful</td>
</tr>
<tr>
<td>-1</td>
<td>Deletion was not (one cutting edge) or not completely (several cutting edges) successful</td>
</tr>
</tbody>
</table>

Data backup

The data are saved during a general tool-data backup (as a component of the D number data sets).
It is advisable to save the sum offsets, in order to allow the current status to be restored in the event of an acute problem. Machine data settings can be made to exclude sum offsets from a data backup (settings can be made separately for "setup offsets" and "sum offsets fine").

**Note**

Sum offsets behave in the same way as D compensations with reference to block search and REPOS. The behavior on Reset and PowerOn can be defined by machine data.

If the setting of the following machine data indicates that the last active tool compensation number (D) is to be activated after PowerOn, the last active DL number is then no longer active:

**MD20110 $MC_RESET_MODE_MASK** (definition of initial control system settings after RESET/TP-End)

### 18.11.4 Examples

**Example 1**

That no compensation and no sum offset will come into effect must be defined during tool change via the machine data:

- **MD20270 $MC_CUTTING_EDGE_DEFAULT=0** (Basic setting of tool cutting edge without programming)
- **MD20272 $MC_SUMCORR_DEFAULT=0** (default setting sum offset without program).

```
T5 M06 ; Tool number 5 is loaded - no compensation active.
D1 DL=3 ; Compensation D1 + sum offset 3 of D1 are activated.
X10
DL=2 ; Compensation D1 + sum offset 2 are activated.
X20
DL=0 ; Sum offset deselection, only compensation D1 is now active.
D2 ; Compensation D2 is activated - the sum offset is not included in the compensation.
X1
DL=1 ; Compensation D2 + sum offset 1 are activated.
X2
D0 ; Compensation deselection
X3
DL=2 ; No effect - DL2 of D0 is zero (same as programming T0 D2).
```

**Example 2**

During tool change it has to be defined that offset **D2** and sum offset **DL=1** are activated via the machine data:
MD20270 $MC_CUTTING_EDGE_DEFAULT=2 (Basic setting of tool cutting edge without programming)

MD20272 $MC_SUMCORR_DEFAULT=1 (default setting sum offset without program)

T5 M06

D1 DL=3

X10

DL=2

X20

DL=0

D2

X1

DL=2

D1

18.11.5 Upgrades for Tool Length Determination

18.11.5.1 Taking the compensation values into account location-specifically and workpiece-specifically

Composition of the effective tool length

For a tool compensation without active kinematic transformation, the effective tool length consists of up to 8 vectors:

- Tool length (geometry) ($TC_DP3 - $TC_DP5)
- Wear ($TC_DP12 - $TC_DP14)
- Tool base dimension (see note) ($TC_DP21 - $TC_DP23)
- Adapter dimension (see note) ($TC_ADPT1 - $TC_ADPT3)
- Total offsets fine ($TC_SCPx3 - $TC_SCPx5)
- Sum offsets coarse or setup offsets ($TC_ECPx3 - $TC_ECPx5)
- Offset vector $I_1$ of toolholder with orientation capability ($TC_CARR1 - $TC_CARR3)$
- Offset vector $I_2$ of toolholder with orientation capability ($TC_CARR4 - $TC_CARR6)$
- Offset vector $I_3$ of toolholder with orientation capability ($TC_CARR15 - $TC_CARR17)$

Note

The tool base dimension and adapter dimension can only be applied as alternatives.
Type of action of the individual vectors

The type of action of the individual vectors or groups of vectors depends on the following further quantities:

<table>
<thead>
<tr>
<th>Influencing quantity</th>
<th>Operating principle</th>
</tr>
</thead>
<tbody>
<tr>
<td>G codes</td>
<td>Active machining plane</td>
</tr>
<tr>
<td>Tool type</td>
<td>Milling tool or turning/grinding tools</td>
</tr>
<tr>
<td>Machine data</td>
<td>Tool management active/not active, toolholder with orient-</td>
</tr>
<tr>
<td>Setting data</td>
<td>Behavior of tool length components when mirroring or</td>
</tr>
<tr>
<td>Toolholder with orientation</td>
<td>Set values of toolholder with orientation capability</td>
</tr>
<tr>
<td>Adapter transformations</td>
<td>Transformed tool compensation values</td>
</tr>
</tbody>
</table>

Distribution over the geometry-axis components

How the three vector components of partial totals of the vectors involved are distributed over the three geometry-axis components is determined by the following quantities:

<table>
<thead>
<tr>
<th>Influencing quantity</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active processing level:</td>
<td>Infeed plane:</td>
</tr>
<tr>
<td>G17 X/Y direction</td>
<td>Z</td>
</tr>
<tr>
<td>G18 Z/X direction</td>
<td>Y</td>
</tr>
<tr>
<td>G19 Y/Z direction</td>
<td>X</td>
</tr>
<tr>
<td>Tool type:</td>
<td>See Section &quot;Tool parameter 1: Tool type (Page 1389)&quot;, Table &quot;Minimum number of</td>
</tr>
<tr>
<td></td>
<td>required tool parameters&quot;</td>
</tr>
<tr>
<td>Toolholder with orientation</td>
<td>See Section &quot;Special handling of tool compensations (Page 1488)&quot; and Section</td>
</tr>
<tr>
<td>transformations</td>
<td>&quot;Setting data (Page 1549)&quot;</td>
</tr>
<tr>
<td>SD42900 $SC_MIRROR_TOOL_LENGTH</td>
<td>References:</td>
</tr>
<tr>
<td>SD42910 $SC_MIRROR_TOOL_WEAR</td>
<td>Function Manual Tool Management</td>
</tr>
<tr>
<td>SD42920 $SC_WEAR_SIGN_CUTPOS</td>
<td></td>
</tr>
<tr>
<td>SD42930 $SC_WEAR_SIGN</td>
<td></td>
</tr>
<tr>
<td>SD42940 $SC_TOOL_LENGTH_CONST</td>
<td></td>
</tr>
<tr>
<td>SD42950 $SC_TOOL_LENGTH_TYPE</td>
<td></td>
</tr>
</tbody>
</table>

The resulting tool orientation always remains parallel to one of the three axis directions X, Y, or Z and exclusively depends on the active machining plane G17–G19, since it has not yet been possible to assign the tool an orientation.

Stepless variation of the tool orientation

The toolholder with orientation capability also enables the tool orientation to be varied steplessly, in addition to providing further offsets or linear expansion fluctuations with the aid of offset vectors \( l_1 \) - \( l_3 \).

For further explanations, see Section "Toolholder with orientation capability (Page 1442)".
Minor operator compensations

Minor compensations, however, must also be modified during the normal production mode. The reasons for this are, for example:

- Tool wear
- Clamping errors
- Temperature sensitivity of the machine:

These compensations are defined as follows:

<table>
<thead>
<tr>
<th>Definition</th>
<th>Wear components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wear</td>
<td>$TC_{DP12} - $TC_{DP14}$,</td>
</tr>
<tr>
<td>Total offsets fine</td>
<td>$TC_{SCPx3} - $TC_{SCPx5}$,</td>
</tr>
<tr>
<td>Sum offsets coarse or setup offsets</td>
<td>$TC_{ECPx3} - $TC_{ECPx5}$</td>
</tr>
</tbody>
</table>

In particular, compensations, which affect the tool length calculation, should be entered in the coordinates used for measurement.

These workpiece-specific compensations can be achieved more simply using the G-code group 56 with the three values $TOWSTD$, $TOWMCS$ and $TOWWCS$ and the setting data:

SD42935 $SC\_WEAR\_TRANSFORM$ (transformation of tool components)

SD42935

Which of the wear components:

- Wear ($TC_{DP12} - TC_{DP14}$)
- Setup offsets or sum offsets coarse ($TC_{ECPx3} - TC_{ECPx5}$)
- Sum offsets fine ($TC_{SCPx3} - TC_{SCPx5}$)

are to be transformed in the transformations:

- Adapter transformation
- Toolholder with orientation capability

are to be or not to be transformed, can be defined via the setting data:

SD42935 $SC\_WEAR\_TRANSFORM$ (transformation of wear values)

With the setting data in its initial state, all wear values are transformed.
The setting data is considered in the following functions:

- Wear values in the machine coordinate system
  Part program instruction: TOWMCS

- Wear values in the workpiece coordinate system
  Part program instruction: TOWWCS

![Diagram of Tool Length Transformation](image)

Figure 18-52 Transformation of wear data dependent on SD42935

### Programming

G-code group 56 can be used to define the following values:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Corrections</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOWSTD</td>
<td>Initial setting value for offsets in tool length</td>
</tr>
<tr>
<td>TOWMCS</td>
<td>Wear values in the machine coordinate system (MCS)</td>
</tr>
<tr>
<td>TOWWCS</td>
<td>Wear values in the workpiece coordinate system (WCS)</td>
</tr>
<tr>
<td>TOWBCS</td>
<td>Wear values in the basic coordinate system (BCS)</td>
</tr>
<tr>
<td>TOWTCS</td>
<td>Wear values in the TCS (Tool Coordinate System) at the toolholder (tool carrier reference point T)</td>
</tr>
<tr>
<td>TOWKCS</td>
<td>Wear values in tool coordinate system for kinematic transformation (KCS) of tool head</td>
</tr>
</tbody>
</table>
Coordinate systems for offsets in tool length

G codes TOWMCS, TOWWCS, TOWBCS, TOWTCS and TOWKCS can be used, e.g. to measure the wear tool length component in five different coordinate systems.

1. Machine coordinate system
2. Basic coordinate system
3. Workpiece coordinate system
4. Tool coordinate system of kinematic transformation
5. Tool coordinate system

The calculated tool length or a tool length component can be represented and read out in one of these coordinate systems using the GETTCOR function (predefined subprogram).

For further explanations, see Section "Read tool lengths, tool length components (Page 1521)."

---

**Figure 18-53 Coordinate system for the evaluation of tool lengths**

---

**18.11.5.2 Functionality of the individual wear values**

**TOWSTD**

Initial setting (default behavior):

- The wear values are added to the other tool length components. The resulting total tool length is then used in further calculations.

In the case of an active toolholder with orientation capability:

- The wear values are subjected to the appropriate rotation.
TOWMCS

Wear data in the MCS (machine coordinate system):

In the case of an active rotation by means of a toolholder with orientation capability:

- The toolholder only rotates the vector of the resultant tool length. Wear is ignored. Then the tool length vector rotated in this way and the wear are added. The wear is not subjected to the rotation.

If no toolholder with orientation capability is active or this does not result in a rotation, TOWMCS and TOWSTD are identical.

Linear transformation

The tool length can be uniquely defined in the MCS only if the MCS is generated by linear transformation from the BCS.

This would be the case when:

- No kinematic transformation is active
- Or orientation transformations (3-axis, 4-axis and 5-axis transformations) are active

TOWWCS

Wear values in WCS (workpiece coordinate system):

- If a toolholder with orientation capability is active, the tool vector is calculated as for TOWMCS, without taking the wear into account.
- The wear data are interpreted in the workpiece coordinate system.

The wear vector in the workpiece coordinate system is converted to the machine coordinate system and added to the tool vector.

TOWBCS

Wear values in BCS (basic coordinate system):

- If a toolholder with orientation capability is active, the tool vector is calculated as for TOWMCS, without taking the wear into account.
- The wear data are interpreted in the workpiece coordinate system.

The wear vector in the basic coordinate system is converted to the workpiece coordinate system and added to the tool vector.

Non-linear transformation

If a non-linear transformation is active, e.g. with TRANSMIT, and the MCS is specified as the desired coordinate system, the BCS is automatically used instead of the MCS.

Toolholder with orientation capability

A table component of the toolholder with orientation capability, if available, is not applied directly to the coordinate systems, unlike a table (or part) component of the kinematic transformation. A rotation described by such a component is represented in a basic frame or system frame and is thus included in the transition from WCS to BCS.

Kinematic transformation
The table (or part) component of the kinematic transformation is described by the transition from BCS to MCS.

**TOWTCS**

Wear values in TCS (tool coordinate system):
- If a toolholder with orientation capability is active, the tool vector is calculated as for TOWMCS, without taking the wear into account.
- The wear data are interpreted in the tool coordinate system.

The wear vector in the TCS (Tool Coordinate System) is converted to the machine coordinate system by way of the tool coordinate system of the kinematic transformation (KCS) and added to the tool vector.

**TOWKCS**

The wear value specifications for the kinematic transformation are interpreted in the associated TCS (Tool Coordinate System).

The wear vector is converted to the machine coordinate system by way of the tool coordinate system of the kinematic transformation and added to the tool vector.

**G code change when a tool is active**

Changing the G code in the group TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, and TOWKCS does not affect an already active tool, and does not become effective until the next tool is selected.

A new G code of this group will also come into effect if it is programmed in the same block, in which a tool is selected.

**Evaluation of individual wear components**

Evaluation of individual wear components (assignment to geometry axes, sign evaluation) is influenced by:
- The active plane
- The adapter transformation
- The five setting data shown in the table below

<table>
<thead>
<tr>
<th>Setting data</th>
<th>Wear components</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD42910 $SC_MIRROR_TOOL_WEAR</td>
<td>TOWSTD</td>
</tr>
<tr>
<td>SD42920 $SC_WEAR_SIGN_CUTPOS</td>
<td>X</td>
</tr>
<tr>
<td>SD42930 $SC_WEAR_SIGN</td>
<td>X</td>
</tr>
<tr>
<td>SD42940 $SC_TOOL_LENGTH_CONST</td>
<td>X</td>
</tr>
<tr>
<td>SD42950 $SC_TOOL_LENGTH_TYPE</td>
<td>X</td>
</tr>
</tbody>
</table>
Note
Wear components which are subjected to an active rotation by an adapter transformation or a toolholder with orientation capability are referred to as non-transformed wear components.

Special features
If TOWMCS or TOWWCS is active, the following setting data does not affect the non-transformed wear components:

SD42920 $SC_WEAR_SIGN_CUTPOS (Sign of wear for tools with cutting edge systems)

The following setting data also does not affect the non-transformed wear components in case of TOWWCS:

SD42910 $SC_MIRROR_TOOL_WEAR (Sign change tool wear when mirroring)

In this case, a possibly active mirroring is already contained in the frame, which is referred to for evaluating the wear components.

On a plane change, the assignment between the non-transformed wear components and the geometry axes is retained, i.e. these are not interchanged as with other length components. The assignment of components depends on the active plane for tool selection.

Example
Let's assume a milling tool is used where only the wear value $TC_DP12 assigned to length L1 is not equal to zero.

If G17 is active, this length is effective in the direction of the Z axis.

This measure always acts in the Z-direction also upon a plane change after the tool selection, when TOWMCS or TOWWCS are active and the bit 1 is set in the setting data:

SD42935 $SC_WEAR_TRANSFORM (transformations for tool components)

If, for example, G18 is active on tool selection, the component is always effective in the Y direction instead.
18.12 Working with tool environments

18.12.1 General

Functions

The current states of tool data can be processed using the functions below, which are generally available:

- Save
- Deletion
- Read
- Modify

A further function can be used to determine information about the assignment of the tool lengths of the active tool to the abscissa, ordinate and applicate.

18.12.2 Saving with TOOLENV

Scope of a tool environment

The TOOLENV memory function is used to save any current states needed for the evaluation of tool data stored in the memory.

The individual data are as follows:

- The active G code of group 6 \((G17, G18, G19)\)
- The active G code of group 56 \((TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, TOWKCS)\)
- The active transverse axis
- Machine data:
  - MD18112 $MN_MM_KIND_OF_SUMCORR (Properties of sum offsets in the TO area)
- Machine data:
  - MD20360 $MC_TOOL_PARAMETER_DEF_MASK (definition of tool parameters).
- Setting data:
  - SD42900 $SC_MIRROR_TOOL_LENGTH (Sign change tool length when mirroring)
- Setting data:
  - SD42910 $SC_MIRROR_TOOL_WEAR (Sign change tool wear when mirroring)
- Setting data:
  - SD42920 $SC_WEAR_SIGN_CUTPOS (Sign of wear for tools with cutting edge systems)
- Setting data:
  - SD42930 $SC_WEAR_SIGN (sign of wear)
- Setting data:
  - SD42935 $SC_WEAR_TRANSFORM (transformations for tool components)
● Setting data:
  SD42940 $SC_LENGTH_CONST (change of tool components on change of planes)

● Setting data:
  SD42950 $SC_TOOL_LENGTH_TYPE (allocation of the tool length components independent of tool type)

● The orientation component of the current complete frame (rotation and mirroring, no work offsets or scales)

● The orientation component and the resulting length of the active toolholder with orientation capability

● The orientation component and the resulting length of an active transformation

● In addition to the data describing the environment of the tool, the T number, D number and DL number of the active tool are also stored, so that the tool can be accessed later in the same environment as the TOOLENV call, without having to name the tool again.

Not in the tool environment

The value of the machine data determines whether the adapter length or the tool base dimension is included in the tool length calculation:

MD18104 $MN_MM_NUM_TOOL_ADAPTER (tool adapter in TO area).

Since a change to this machine data only takes effect after Power On, it is not saved in the tool environment.

Note

Resulting length of toolholders with orientation capability and transformations:

Both toolholders with orientation capability and transformations can use system variables or machine data, which act as additional tool length components, and which can be subjected partially or completely to the rotations performed. The resulting additional tool length components must also be stored when TOOLENV is called, because they represent part of the environment, in which the tool is used.

Adapter transformation:

The adapter transformation is a property of the tool adapter and thus of the complete tool. It is, therefore, not part of a tool environment, which can be applied to another tool.

By saving the complete data necessary to determine the overall tool length, it is possible to calculate the effective length of the tool at a later point in time, even if the tool is no longer active or if the conditions of the environment (e.g., G codes or setting data) have changed. Similarly, the effective length of different tool can be calculated assuming that it would be used under the same conditions as the tool, for which the status was saved.

TOOLENV function

Saving a tool environment
The `TOOLENV` function is a predefined subprogram. It must, therefore, be programmed in a separate block.

Syntax:

```
Status = TOOLENV(_NAME)
```

Value/parameter:

- **Status**
  - INT
  - 0: Function OK
  - -1: No memory reserved for tool environments:
    - MD18116 $MN_MM_NUM_TOOL_ENV = 0 (number of tool environments in TO area).
    - i.e. the "tool environments" functionality is not available.
  - -2: No more free memory locations for tool environments available.
  - -3: Null string illegal as name of a tool environment.
  - -4: No parameter (name) specified.

- **_NAME**
  - STRING
  - Name, under which the current data set is stored.
  - If a data set of the same name already exists, it is overwritten. In this case, the status is 0.

### 18.12.3 Delete tool environment

**DELTOOLENV function**

This function can be used to delete sets of data used to describe tool environments. Deletion means that the set of data stored under a particular name can no longer be accessed (an access attempt triggers an alarm).

The `DELTOOLENV` function is a predefined subprogram.

It must, therefore, be programmed in a separate block.

Syntax:

There are two call formats:

```
Status = DELTOOLENV()
Status = DELTOOLENV(_NAME)
```

Value/parameter:

- **Status**
  - INT
  - 0: Function OK
-1: No memory reserved for tool environments:
   MD18116 $MN_MM_NUM_TOOL_ENV = 0 (number of tool environments in TO area).
   i.e. the "tool environments" functionality is not available.

-2: A tool environment with the specified name does not exist.

_NAME STRING
Name of data set to be deleted

The first call format deletes all data sets.
The second call format deletes the data set with the specified name.
Data sets can only be deleted using the DELTOOLENV command, by an INITIAL.INI download or by a cold start (NCK powerup with default machine data). There are no further automatic deletion operations (e.g., on RESET).

18.12.4 How many environments and which ones are saved?

$P_TOOLEVN
This system variable returns the number of available data sets for describing tool environments. (Data sets defined by TOOLENV and not yet deleted.)
The value range is from 0 to machine data:
MD18116 $MN_MM_NUM_TOOL_ENV (number of tool environments in TO area).
This system variable can be accessed even if no tool environments are possible (MD18116 = 0). In this case, the return value is 0.

Syntax:
\_N = $P_TOOLEVN

Data type:
\_N INT
Number of defined TOOLENV

$P_TOOLENV
This system variable returns the number of the nth data set for describing a tool environment.
The assignment of numbers to data sets is not fixed, but can be changed as a result of deleting or creating data sets. The data sets are numbered internally. The range is from 1 to $P_TOOLEVN.

Syntax:
\_NAME = $P_TOOLENV[i]
18.12.5 Read T, D, DL from a tool environment

GETTENV function

The **GETTENV** function is used to read the T, D and DL numbers stored in a tool environment. The **GETTENV** function is a predefined subprogram. It must, therefore, be programmed in a separate block.

Syntax:

```
Status = GETTENV(_NAME, _TDDL)
```

Value/parameter:

- **Status**
  - **INT**
  - 0: Function OK
  - -1: No memory reserved for tool environments:
    ```
    MD18116 $MN_MM_NUM_TOOL_ENV = 0 (number of tool environments in TO area).
    ```
    i.e. the “tool environments” functionality is not available.
  - -2: A tool environment with the name specified in _NAME does not exist.

- **_NAME**
  - **STRING**
  - Name of the tool environment, from which the T, D and DL numbers can be read

- **_TDDL[3]**
  - **INT**
  - This integer array contains:
    - in "_TDDL[0]" the T number of the tool,
    - in "_TDDL[1]" the D number of the tool,
    - in "_TDDL[2]" the DL number of the tool,
    whose tool environment in the data set is stored with the name "_NAME".
It is possible to omit the first parameter in the `GETTENV` function call (e.g., `GETTENV(_, _TDDL)` or to pass a null string as the first parameter (e.g., `GETTENV("", _TDDL)`). In both of these two special cases, the T, D and DL numbers of the active tool are returned in `_TDDL`.

18.12.6 Read tool lengths, tool length components

GETTCOR function

The `GETTCOR` function is used to read out tool lengths or tool length components. The parameters can be used to specify, which components are considered, and the conditions, under which the tool is used.

The `GETTCOR` function is a predefined subprogram. It must, therefore, be programmed in a separate block.

Syntax:

```
Status = GETTCOR(_LEN, _COMP, _STAT, _T, _D, _DL)
```

All parameters can be omitted with the exception of the first parameter (_LEN).

Value/parameter:

**Status**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Function OK</td>
</tr>
<tr>
<td>-1</td>
<td>No memory reserved for tool environments: MD18116 $MN_MM_NUM_TOOL_ENV = 0 (number of tool environments in TO area). i.e. the &quot;tool environments&quot; functionality is not available.</td>
</tr>
<tr>
<td>-2</td>
<td>A tool environment with the name specified in _STAT does not exist.</td>
</tr>
<tr>
<td>-3</td>
<td>Invalid string in parameter _COMP. Causes of this error can be invalid characters or characters programmed twice.</td>
</tr>
<tr>
<td>-4</td>
<td>Invalid T number</td>
</tr>
<tr>
<td>-5</td>
<td>Invalid D number</td>
</tr>
<tr>
<td>-6</td>
<td>Invalid DL number</td>
</tr>
<tr>
<td>-7</td>
<td>Attempt to access non-existent memory module</td>
</tr>
<tr>
<td>-8</td>
<td>Attempt to access a non-existent option (programmable tool orientation, tool management).</td>
</tr>
<tr>
<td>-9</td>
<td>The _COMP string contains a colon (identifier for the specification of a coordinate system), but it is not followed by a valid character denoting the coordinate system.</td>
</tr>
</tbody>
</table>

**_LEN[11]**

Result vector

The vector components are arranged in the following order:

- Tool type (LEN[0])
- Length of cutting edge (LEN[1])
The coordinate system defined in _COMP and _STAT is used as the reference coordinate system for the length components. If no coordinate system is defined in _COMP, the tool lengths are represented in the machine coordinate system.

The assignment of the abscissa, ordinate and applicate to the geometry axes depends on the active plane in the tool environment, i.e. with G17, the abscissa is parallel to X, with G18 it is parallel to Z, etc.

Components LEN[6] to LEN[10] contain the additional parameters, which can be used to specify the geometry description of a tool (e.g. $TC_DP7 to $TC_DP11 for the geometry and the corresponding components for wear or sum and setup offsets).

These 5 additional elements and the tool radius are only defined for components E, G, S, and W. Their evaluation does not depend on _STAT. The corresponding values in LEN[5] to LEN[10] can thus only be not equal to zero if at least one of the four specified components is involved in the tool length calculation. The remaining components do not influence the result. The dimensions refer to the control's basic system (inch or metric).

_COMP STRING
This string consists of two substrings, which are separated from one another by a colon. The individual characters (letters) of the first substring identify the tool length components to be taken into account when calculating the tool length.

The second substring identifies the coordinate system, in which the tool length is to be output. It consists of only one single relevant character.

The order of the characters in the strings, and their notation (upper or lower case), is arbitrary. Any number of blanks or white spaces can be inserted between the characters.

The letters in the substrings cannot be programmed twice. The meanings in the first substring are as follows:

-: (Minus symbol, only allowed as first character): The complete tool length is calculated, minus the components specified in the next string.
C: Adapter or tool base dimension (whichever of the two alternative components is active for the tool in use)
E: Setup offsets
G: Geometry
K: Kinematic transformation (is only evaluated for generic 3, 4 and 5-axis transformation)
S: Sum offsets
T: Toolholder with orientation capability
W: Wear
If the first substring is empty (except for white spaces), the complete tool length is calculated allowing for all components. This applies even if the _COMP parameter is not specified.

An optional programmable colon must be followed by a single character specifying the coordinate system, in which the tool length components are to be evaluated. If no coordinate system is specified, the evaluation is performed in the MCS (machine coordinate system). If any rotations are to be taken into account, they are specified in the tool environment defined in _STAT.

The characters have the following significance:

- **B**: Basic coordinate system (BCS)
- **K**: Tool coordinate system of kinematic transformation (KCS)
- **M**: Machine coordinate system (MCS)
- **T**: Tool coordinate system (TCS)
- **W**: Workpiece coordinate system (WCS)

**_STAT STRING**

Name of the data set for describing a tool environment.

If the value of this parameter is the null string (""") or is not specified, the current status is used.

**_T INT**

Internal T number of tool

If this parameter is not specified, or if its value is 0, the tool stored in _STAT is used.

If the value of this parameter is -1, the T number of the active tool is used. It is also possible to specify the number of the active tool explicitly.

**Note**

If _STAT is not specified, the current status is used as the tool environment. Since _T = 0 refers to the T number saved in the tool environment, the active tool is used in that environment, i.e. parameters _T = 0 and _T = -1 have the same meaning in this special case.

**_D INT**

Cutting edge of the tool. If this parameter is not specified, or if its value is 0, the D number used is based on the source of the T number. If the T number from the tool environment is used, the D number of the tool environment is also read, otherwise the D number of the currently active tool is read.

**_DL INT**

Number of the local compensation. If this parameter is not specified, the DL number used is based on the source of the T number. If the T number from the tool environment is used, the D number of the tool environment is also read, otherwise the D number of the currently active tool is read.
Any rotations and component exchanges initiated by the adapter transformation, toolholder with orientation capability and kinematic transformation, are part of the tool environment. They are thus always performed, even if the corresponding length component is not supposed to be included. If this is undesirable, tool environments must be defined, in which the corresponding transformations are not active. In many cases (i.e. any time a transformation or toolholder with orientation capability is not used on a machine), the data sets stored for the tool environments automatically fulfill these conditions, with the result that the user does not need to make special provision.

**MD20360 $MC_TOOL_PARAMETER_DEF_MASK**

The two least significant bits of this machine data specify how the wear (bit 0) and tool length (bit 1) are to be evaluated if a diameter axis is used for turning and grinding tools.

If the bits are set, the associated entry is weighted with the factor 0.5. This weighting is reflected in the tool length returned by GETTCOR.

**Example:**

MD20360 $MC_TOOL_PARAMETER_DEF_MASK = 3 (definition of tool parameters).

MD20100 $MC_DIAMETER_AX_DEF=“X” (Geometry axis with face axis function)

X is diameter axis (standard turning machine configuration):

```
N30  $TC_DP1[1.1] =  500
N40  $TC_DP2[1.1] =  2
N50  $TC_DP3[1.1] =  3.0  ;  Geometry L1
N60  $TC_DP4[1,1]=  4.0
N70  $TC_DP5[1,1]=  5.0
N80  $TC_DP12[1,1]=  12.0  ;  Wear L1
N90  $TC_DP13[1,1]=  13.0
N100 $TC_DP14[1,1]=  14.0
N110 t1 d1 g18
N120  r1 = GETTCOR(_LEN, "GW")
N130  r3 = _LEN[2]  ; 17.0 (- 4.0 + 13.0)
N140  r4 = _LEN[3]  ;  7.5 (- 0.5 * 3.0 + 0.5 * 12.0)
N150  r5 = _LEN[4]  ; 19.0 (- 5.0 + 14.0)
N160  m30
```

**Kinematic transformation, toolholder with orientation capability**

If a toolholder with orientation capability is taken account of during the tool length calculation, the following vectors are included in that calculation:

<table>
<thead>
<tr>
<th>Type</th>
<th>Vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>$l_1$ and $l_2$</td>
</tr>
<tr>
<td>T</td>
<td>$l_1$, $l_2$ and $l_3$</td>
</tr>
<tr>
<td>P</td>
<td>Tool length is not influenced by the toolholder with orientation capability.</td>
</tr>
</tbody>
</table>
In generic 5-axis transformation, the following machine data are included in the tool length calculation for transformer types 24 and 56:

<table>
<thead>
<tr>
<th>Transformer type</th>
<th>Machine data</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>MD24550/24650 $MC_TRAFO5_BASE_TOOL_1/2</td>
</tr>
<tr>
<td></td>
<td>MD24560/24660 $MC_TRAFO5_JOINT_OFFSET_1/2</td>
</tr>
<tr>
<td></td>
<td>MD24558/24658 $MC_TRAFO5_PART_OFFSET_1/2</td>
</tr>
<tr>
<td>56</td>
<td>MD24550/24650 $MC_TRAFO5_BASE_TOOL_1/2</td>
</tr>
<tr>
<td></td>
<td>MD24560/24660 $MC_TRAFO5_JOINT_OFFSET_1/2</td>
</tr>
</tbody>
</table>

Transformation type 56 corresponds to type M for a toolholder with orientation capability.

With this 5-axis transformation in the software versions used up to now, the following vector is equivalent to the sum of the two vectors $l_1$ and $l_3$ for a toolholder with orientation capability type M.

MD24560/24660 $MC\_TRAFO5\_JOINT\_OFFSET\_1/2$ (vector of kinematic offset of the first/second 5-axis transformation in the channel)

Only the sum is relevant for the transformation in both cases. The way, in which the two individual components are composed, is insignificant. When calculating the tool length, however, it is relevant which component is assigned to the tool and which is assigned to the tool table.

This explains the introduction of new machine data:

MD24558/24658 $MC\_TRAFO5\_JOINT\_OFFSET\_PART\_1/2$ (vector kinematic offset in table).

It is equivalent to the vector $l_3$.

The following machine data no longer corresponds to the sum of $l_1$ and $l_3$, but only to vector $l_1$.

MD24560/24660 $MC\_TRAFO5\_JOINT\_OFFSET\_1/2$ (vector of kinematic offset of the first 5-axis transf. in the channel).

The new response is identical to the current response, if the following machine data equals zero:

MD24558/24658 $MC\_TRAFO5\_JOINT\_OFFSET\_PART\_1/2$ (vector kinematic offset in table).

**GETTCOR examples**

<table>
<thead>
<tr>
<th>GETTCOR(_LEN):</th>
<th>Calculates the tool length of the currently active tool in the machine coordinate system allowing for all components.</th>
</tr>
</thead>
<tbody>
<tr>
<td>GETTCOR(_LEN; “CGW : W”):</td>
<td>Calculates the tool length for the active tool, consisting of the adapter or tool base dimension, geometry and wear. Further components, such as toolholder with orientation capability or kinematic transformation, are not considered. The workpiece coordinate system is used for the output.</td>
</tr>
</tbody>
</table>
GETTCOR(_LEN, "-K :B"): Calculates the complete tool length of the active tool without allowing for the length components of an active kinematic transformation. Output in the basic coordinate system.

GETTCOR(_LEN, "-M", "Testenv1", ,3): Calculates the complete tool length in the machine coordinate system for the tool stored in the tool environment named “Testenv1”. The calculation is performed for cutting edge number D3, regardless of the cutting edge number stored.

Compatibility

The GETTCOR function is used in conjunction with the TOOLENV and SETTCOR functions to replace parts of the functionality, which were previously implemented externally in the measuring cycles.

Only some of the parameters, which actually determine the effective tool length, were implemented in the measuring cycles. The above functions can be configured to reproduce the behavior of the measuring cycles in relation to the tool length calculation.

18.12.7 Changing tool components

SETTCOR function

The SETTCOR function is used to change tool components taking into account all general conditions that can be involved when evaluating the individual components.

The SETTCOR function is a predefined subprogram. It must, therefore, be programmed in a separate block.

Note

Regarding the terminology: If in the following, in conjunction with the tool length, tool components are involved, then the components considered from a vectorial perspective are meant, which make up the complete tool length, e.g. geometry or wear. Such a component comprises three individual values (L1, L2, L3), which are called coordinate values in the following.

The tool component "geometry" therefore comprises three coordinate values $TC_DP3 to $TC_DP5.

Syntax:

\[
\text{Status} = \text{SETTCOR}(_\text{CORVAL}, _\text{COMP}, _\text{CORCOMP}, _\text{CORMODE}, _\text{GEOAX}, _\text{STAT}, _\text{T}, _\text{D}, _\text{DL})
\]

With the exception of the first two parameters (_CORVAL and _COMP) all of the parameters can also be omitted.

Value/parameter:

Status INT
0: Function OK
-1: No memory reserved for tool environments:
   MD18116 $MN_MM_NUM_TOOL_ENV = 0 (number of tool environments in TO area).
   i.e. the "tool environments" functionality is not available.
-2: A tool environment with the name specified in _STAT does not exist.
-3: Invalid string in parameter _COMP.
   Causes of this error can be invalid characters or characters programmed twice.
-4: Invalid T number.
-5: Invalid D number.
-6: Invalid DL number.
-7: Attempt to access a non-existent memory module.
-8: Attempt to access a non-existent option (programmable tool orientation, tool management).
-9: Illegal numerical value for the _CORCOMP parameter.
-10: Illegal numerical value for the _CORMODE parameter.
-11: The contents of the _COMP and _CORRCOMP parameters are contradictory.
-12: The contents of the _COMP and _CORRMODE parameters are contradictory.
-13: The content of the _GEOAX parameter does not designate a geometry axis.
-14: Write attempt to a non-existent setup offset.

_CORVAL[3] REAL array
Designates the offset vector.
In this case, in the workpiece coordinate system (WCS) defined using _STAT the following are assigned:
- _CORVAL[0] of the abscissa
- _CORVAL[1] of the ordinate
- _CORVAL[2] of the applicate
If only one tool component is to be corrected (i.e. no vectorial correction, see parameter _CORMODE), the correction value is always in _CORVAL[0], independent of the axis on which it acts. The contents of the other two components are then not evaluated.
If _CORVAL or a component of _CORVAL refers to the transverse axis, then the data is evaluated as radius dimension. This means that a tool is e.g. "longer" by the specified dimension; this correspondingly results in a change to the workpiece diameter that is twice as large.
The dimensions refer to the basic system (inch or metric) of the control system.

_COMP STRING
String that comprises either one or two characters. The first or only character for the 1st component (Val_1) and the second character for the 2nd component (Val_2), which are processed according to the subsequent parameters _CORCOMP and _CORMODE.
The notation of the characters in the string (upper or lower case) is arbitrary. Any number of spaces or tabs (white spaces) can be inserted.

The individual meanings are as follows:

C: Adapter or tool base dimension (whichever of the two alternative components is active for the tool in use)
E: Setup offsets
G: Geometry
S: Sum offsets
W: Wear

_CORCOMP INT
This parameter specifies the component(s) of the two data sets that are to be described. If this parameter is not specified then its value is 0.

Meaning of the numerical values:

0: The offset value _CORVAL[0] refers to the geometry axis transferred in parameter _GEOAX in the workpiece coordinate system, i.e. the offset value must be calculated in the designated tool components so that, taking account all the parameters that can influence the tool length calculation, as a result, a change of the total tool length by the specified value in the specified axis direction is obtained.

This change should be achieved by the correction of the component specified in _COMP and the symbolic algorithm specified in _CORMODE (see the following parameters). The resulting correction can therefore have an effect on all three axis components.

1: Like 0, however, vectorial. The content of vector _CORVAL refers to abscissa, ordinate, applicate in the workpiece coordinate system (WCS).

The subsequent parameter _GEOAX is not evaluated.

2: Vectorial offset, i.e. L1, L2 and L3 can change simultaneously.

In contrast to the versions from 0 and 1, the offset values contained in _CORVAL refer to the coordinates of Val1 components (see following parameter _CORMODE) of the tool.

Any possible inclination of an existing tool compared with the workpiece coordinate system has no influence on the offset.

3 - 5: Correction of tool lengths L1 to L3 ($TC_DP3 to $TC_DP5) or the corresponding values for wear, setting up or additive offsets.

The offset value is contained in _CORVAL[0]. It is measured in the coordinates of the Val1 component (see following parameter _CORMODE) of the tool. Any possible inclination of an existing tool compared with the workpiece coordinate system has no influence on the offset.

6: Correction of the tool radius ($TC_DP6) or the corresponding values for wear, setting up or additive offsets.

7 – 11: Correction of $TC_DP7 to $TC_DP11 or the corresponding values for wear, setting up or additive offsets. These parameters are treated just like the tool radius.
_CORMODE  INT
This parameter specifies the type of write operation to be executed.
If this parameter is not specified then its value is 0.
Meaning of the ones location:

0: \( \text{Val}_{1\text{new}} = \_\text{CORVAL} \)
1: \( \text{Val}_{1\text{new}} = \text{Val}_{1\text{old}} + \_\text{CORVAL} \)
2: \( \text{Val}_{1\text{new}} = \_\text{CORVAL} \)
    \( \text{Val}_{2\text{new}} = 0 \)
3: \( \text{Val}_{1\text{new}} = \text{Val}_{1\text{old}} + \text{Val}_{2\text{old}} + \_\text{CORVAL} \)
    \( \text{Val}_{2\text{new}} = 0 \)

The notation \( \text{Val}_{1\text{old}} + \text{Val}_{2\text{old}} \) is symbolic. If the two components (due to the status of \_STAT) are evaluated in different ways, i.e. if a rotation is effective between the two components, then \( \text{Val}_{2\text{old}} \) is transformed prior to addition so that the resulting tool length after deleting \( \text{Val}_{2\text{new}} \) and prior to the addition of \_CORVAL remains unchanged.

\_CORVAL always refers to \( \text{Val}_{1\_\text{CORVAL}} \) is a value that is always measured in the workpiece coordinate system (WCS). It is therefore already transformed with respect to the tool components, in which it should be calculated. Therefore, it cannot be directly calculated together with the saved value, but must be transformed back prior to adding to \( \text{Val}_{1} \) or \( \text{Val}_{2} \).

This can mean that the offset acts on an axis different than the one defined by \_CORCOMP – or that it acts on several axes.

For the case \_CORRCOMP = 0, i.e. if \_CORVAL does not contain a vector, but only an individual value, then the described operations are executed in the coordinates in which \_CORVAL was measured (WCS). In particular, this also applies to setting \( \text{Val}_{2\text{new}} \) to zero in versions 2 and 3. This result is then transformed back into the coordinates of the tool.

This can mean that none of the coordinate values to be set to zero (L1, L2, L3) become zero, or coordinate values, that were previously zero, are now not equal to zero. However, if the corresponding operations are successively executed for all three geometry axes, then all three coordinate values of the components to be deleted are always zero. If the tool is not rotated with respect to the workpiece coordinate system or is rotated so that all tool components remain parallel to the coordinate axes (axis exchange operations), then this also ensures that only one tool coordinate changes.

The successive execution of the same operation (_CORMODE) with _CORCOMP = 0 for all three coordinate axes in any sequence is identical with the single execution of the same operation with _CORCOMP = 1.

_T  INT
Internal T number of the tool. If this parameter is not specified or if its value is 0, then the tool stored in _STAT is used. If the value of this parameter is -1, the T number of the active tool is used. It is also possible to explicitly specify the number of the active tool.
**Note**

If \_STAT is not specified, the actual status is used as the tool environment. Since \_T = 0 refers to the T number saved in the tool environment, the active tool is used in this environment, i.e. parameters \_T = 0 and \_T = -1 have the same meaning in this special case.

\_D \hspace{1cm} \text{INT}

Cutting edge of the tool. If this parameter is not specified, or if its value is 0, the D number used is based on the source of the T number. If the T number from the tool environment is used, the D number of the tool environment is also read, otherwise the D number of the currently active tool is read.

\_DL \hspace{1cm} \text{INT}

Number of the offset dependent on the location. If this parameter is not specified, the DL number used is based on the source of the T number. If the T number from the tool environment is used, the D number of the tool environment is also read, otherwise the D number of the currently active tool is read. If T, D and DL specify a tool without location-dependent offsets, no additive or setting-up offsets may be specified in parameter \_COMP (error code in "Status").

Not all possible combinations of the three parameters \_COMP, \_CORCOMP and \_CORMODE are sensible. For example, algorithm 3 in \_CORCOMP requires that two characters are specified in \_COMP. If an invalid parameter combination is specified, then a corresponding error code is returned in the status.

**Note**

Calculating the tool length depending on machine data MD20360

The two least significant bits of this machine data specify how the wear (bit 0) and tool length (bit 1) are to be evaluated if a diameter axis is used for turning and grinding tools. If the appropriate bits are set, then for the tool length calculation, a factor of 0.5 is applied to the associated entry. The correction using \texttt{SETTCOR} is executed so that the total effective tool length change is equal to the value transferred in \_CORVAL.

The correction of the components, whose length is evaluated with a factor of 0.5 due to the following machine data for the length calculation, must be realized using twice the transferred value:

\begin{verbatim}
MD20360 $MC_TOOL_PARAMETER_DEF_MASK (definition of tool parameters)
\end{verbatim}

**Example 1**

\begin{verbatim}
N10   def real \_CORVAL[3]
N20   $TC_DP1[1,1] = 120 ; Milling tool
N30   $TC_DP3[1,1] = 10.0 ; Geometry L1
\end{verbatim}
18.13 Tool lengths L1, L2, L3 assignment: LENTOAX

LENTOAX function

The "LENTOAX" function provides information about the assignment of tool lengths L1, L2 and L3 of the active tool to the abscissa, ordinate and applicate. The assignment of abscissa, ordinate and applicate to the geometry axes is affected by frames and the active plane \(G_{17}\) - \(G_{19}\).

Only the geometry component of a tool ($TC_{DP3}[x,y]$ to $TC_{DP5}[x,y]$) is considered, i.e. a different axis assignment for other components (e.g. wear) has no effect on the result.

The "LENTOAX" function is a predefined subprogram. It must, therefore, be programmed in a separate block.

**Syntax:**

\[
\text{Status} = \text{LENTOAX}(_{AXIND}, _{MATRIX}, _{COORD})
\]
The first two parameters must always be programmed; the last parameter can be omitted.

**Value/parameter:**

- **Status**
  - **0:** Function OK, information in \_AXIND sufficient for description (all tool length components are parallel to the geometry axes).
  - **1:** Function is OK, however, the content of \_MATRIX must be evaluated for a correct description (the tool length components are not parallel to the geometry axes).
  - **-1:** Invalid string in parameter \_COORD.
  - **-2:** No tool active.

- **\_AXIND[3]**
  - **array**
  - Indices 0 to 2 are assigned to the abscissa (0), ordinate (1) and applicate (2) (e.g. \_AXIND[0] contains the number of the tool length components, which are effective in the direction of the abscissa).
  - The content has the following significance:
    - **0:** Assignment exists (axis does not exist)
    - **1 to 3:** Number of the length effective in the corresponding coordinate axis. The sign is negative if the tool length component is pointing in the negative coordinate direction.
    - **-1 to -3:** Number of the length effective in the corresponding coordinate axis. The sign is negative if the tool length component is pointing in the negative coordinate direction.

- **\_MATRIX[3][3]**
  - **array**
  - Matrix which represents the vector of the tool lengths \((L_1=1, L_2=1, L_3=1)\) to the vector of the coordinate axes (abscissa, ordinate, applicate), i.e. the tool length components are assigned to the **columns** in the order \(L_1, L_2, L_3\) and the axes are assigned to the **lines** in the order abscissa, ordinate, applicate.
  - All elements are always valid in the matrix, even if the geometry axis belonging to the coordinate axis is not available, i.e. if the corresponding entry in \_AXIND is 0.

- **\_COORD**
  - **STRING**
  - Specifies the coordinate system used for the assignment.
    - **MCS or M:** The tool length is represented in the machine coordinate system.
    - **BCS or B:** The tool length is represented in the basic coordinate system.
    - **WCS or W:** The tool length is represented in the workpiece coordinate system (default).
    - **KCS or K:** The tool length is represented in the tool coordinate system of the kinematic transformation.
    - **TCS or T:** The tool length is represented in the tool coordinate system.
  - The notation of the characters in the string (upper or lower case) is arbitrary.
Further explanations

If the tool length components are parallel to the geometry axes, the axis indices assigned to length components \(L_1\) to \(L_3\) are returned in the \_AXIND array.

If a tool length component points in the negative axis direction, the associated axis index contains a minus sign. In this case, the return value \(\text{status}\) is 0. If an axis does not exist, the associated return value is 0. The assignment can also be read from the \_MATRIX parameter. Six of the nine matrix elements are then zero, and three elements contain the value +1 or -1.

Note

In the TCS, all tool length components are always parallel or antiparallel to the axes.

The components can only be antiparallel when mirroring is active and the following setting data is activated:

\[
\text{SD42900 } \$\text{SC\_MIRROR\_TOOL\_LENGTH} \text{ (sign change tool length when mirroring)}
\]

If not all length components are parallel or antiparallel to the geometry axes, the index of the axis, which contains the largest part of a tool length component, is returned in \_AXIND. In this case (if the function does not return an error for a different reason), the return value is 1. The mapping of tool length components \(L_1\) to \(L_3\) onto geometry axes 1 to 3 is then described completely by the contents of the 3rd parameter \_MATRIX.

The \_COORD parameter can be used to specify, which coordinate system is to be used for the geometry axes. If the \_COORD parameter is not specified (notation \(\text{LENTOAX(AXIND, _MATRIX)}\)), the WCS is used (default).

Example:

Standard situation: milling tool with \(\text{G17}\)
\(L_1\) applies in \(Z\) (applicate), \(L_2\) applies in \(Y\) (ordinate), \(L_3\) applies in \(X\) (abscissa).

Function call in the form:

\[
\text{Status } = \text{LENTOAX(_AXIND, _MATRIX, "WCS")}
\]

The result parameter \_AXIND contains the values:

\[
\_AXIND[0] = 3 \\
\_AXIND[1] = 2 \\
\_AXIND[2] = 1
\]

Or, in short: (3, 2, 1)

In this case, the associated matrix \_MATRIX is:

\[
\begin{bmatrix}
1 \\
1 \\
1
\end{bmatrix}
\]

A change from \(\text{G17}\) to \(\text{G18}\) or \(\text{G19}\) does not alter the result, because the assignment of the length components to the geometry axes changes in the same way as the assignment of the abscissa, ordinate and applicate.
A frame rotation of Z through 60 degrees is now programmed with G17 active, e.g. \( \text{rot } 260 \). The direction of the applicate (Z direction) remains unchanged; the main component of \( L_2 \) now lies in the direction of the new X axis; the main component of \( L_1 \) now lies in the direction of the negative Y axis. The return axis is thus 1, and \( \_AXIND \) contains the values \((2, -3, 1)\).

In this case, the associated matrix \( \_MATRIX \) is:

\[
\begin{pmatrix}
\sin(\theta) & \cos(\theta) \\
\cos(\theta) & -\sin(\theta) \\
0 & 0 & 1
\end{pmatrix}
\]

**Note**

For further information to the above mentioned coordinate systems, please refer to:

**References:**
Programming Manual, Job Planning; Tool Offsets:

### 18.14 Supplementary conditions

#### 18.14.1 Flat D number structure

**Grinding tools**

Grinding tools (tool types 400-499) cannot be defined using the simple tool management structure (flat D numbers).

**Block search**

T number output to PLC triggers a synchronization process in the NCK: with absolute, indirect D programming, the PLC returns the D values via NC/PLC interfaces. The NCK waits until the output of a T number is followed by a response from the PLC: "I have written the D number". With block search without calculation, this process of synchronization must be deactivated until the first valid T number has been output again. That means that the NCK must not wait on D programming.

**Note**

At what point the auxiliary functions can be output to PLC after block search is complete, can be controlled with the machine data:

\$MC\_AUXFU\_AT\_BLOCK\_SEARCH\_END (auxiliary function output after block search)

Automatic on end or on NC start.
The (only) writable variable $A_MONIFACT, which is defined here, is stored by main-run data. Since the write process takes place synchronously to the main run, no special measures are required for Reorg.

18.14.2 SD42935 expansions

SD42935

Which of the wear components are to be transformed and which are not to be transformed in conjunction with the functions TOWMCS and TOWWCS can be defined via the setting data:

SD42935 $SC_WEAR_TRANSFORM (transformation of wear values)

18.15 Examples

18.15.1 Toolholder with orientation capability

18.15.1.1 Example: Toolholder with orientation capability

Requirement

The following example uses a toolholder, which is described fully by a rotation about the Y axis. It is therefore sufficient to enter only one value to define the rotary axis (block N20).

Blocks N50 to N70 describe an end mill with radius 5 mm and length 20 mm.

Block N90 defines a rotation of 37 degrees about the Y axis.

Block N120 activates the tool radius compensation and all settings are made to describe the compensation in the following blocks with a rotation of 37 degrees about the Y axis.

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>; Definition of toolholder 1</td>
</tr>
<tr>
<td>N20</td>
<td>$TC_CARR8[1] = 1</td>
</tr>
<tr>
<td>N30</td>
<td></td>
</tr>
<tr>
<td>N40</td>
<td>; Definition of tool-compensation memory T1/D1</td>
</tr>
<tr>
<td>N50</td>
<td>$TC_DP1[1,1] = 120</td>
</tr>
<tr>
<td>N60</td>
<td>$TC_DP3[1,1] = 20</td>
</tr>
<tr>
<td>N70</td>
<td>$TC_DP6[1,1] = 5</td>
</tr>
<tr>
<td>N80</td>
<td></td>
</tr>
<tr>
<td>N90</td>
<td>ROT Y37</td>
</tr>
<tr>
<td>N100</td>
<td></td>
</tr>
</tbody>
</table>

W1: Tool offset

18.15 Examples
18.15.1.2 Example of toolholder with orientation capability with rotary table

Use of the MOV command

For use of the MOV command it is assumed that the program is running on a 5-axis machine, on which the tool rotates about the Y axis in case of a rotation of the B axis:

N10 TRAORI()
N20 X0 X0 Z0 B45 F2000 ; Setting the tool orientation
N30 MOV=-10 ; Infeed movement 10 mm in tool direction
; (under 45 degrees in the Y-Z plane)
N40 MOV=AC(20) ; Retraction in tool direction at distance of
; 20 mm from the zero point

Machine with rotary table

Complete definition for the use of a toolholder with orientation capability with rotary table:

N10 $TC_DP1[1,1]=120
N20 $TC_DP3[1,1]=13 ; Tool length 13 mm

; Definition of toolholder 1:
N30 $TC_CARR1[1] = 0 ; X component of 1st offset vector
N40 $TC_CARR2[1] = 0 ; Y component of 1st offset vector
N50 $TC_CARR3[1] = 0 ; Z component of 1st offset vector
N60 $TC_CARR4[1] = 0 ; X component of 2nd offset vector
N70 $TC_CARR5[1] = 0 ; Y component of 2nd offset vector
N80 $TC_CARR6[1] = -15 ; Z component of 2nd offset vector
N90 $TC_CARR7[1] = 1 ; X component of 1st axis
N100 $TC_CARR8[1] = 0 ; Y component of 1st axis
N110 $TC_CARR9[1] = 0 ; Z component of 1st axis
N120 $TC_CARR10[1] = 0 ; X component of 2nd axis
The definition of the toolholder with orientation capability is given in full. The components which contain the value 0 need not actually be given, as they are preset to zero in any case.

The toolholder is activated in N270.

As $TC_CARR21$ and $TC_CARR22$ refer to the machine axes A and B and TCOABS is active, the values in $TC_CARR13$ and $TC_CARR14$ are ignored, i.e. the axis position A0 B45 is used for the rotation.

The rotation of the 4th offset vector (length 15 mm in Z direction) around the B axis causes an offsetting of the zero point by X10.607 [= 15 * sin(45)] and Z-4.393 [= -15 * (1 - cos(45))]. This zero offset is taken into account by an automatically written basic or system frame so that the position X10.607 Y10.000 Z8.607 is approached. In the Z direction the tool selection leads to an additional offset of 13 mm; the Y component is not affected by the table rotation.
N280 defines a rotation in accordance with the rotation of the table of the toolholder with orientation capability. The new X direction thus points in the direction of the bisecting line in the 4th quadrant, the new Z axis in the direction of the bisecting line in the 1st quadrant.

The zero point is approached in N290, i.e. the machine position X10.607 Y0 Z-4.393, since the position of the zero point is not changed by the rotation.

N300 traverses in Y to the position Y33.000, since G18 is active and the Y component is not affected by the active frame. The X and Z positions remain unchanged.

The position X17.678 Y0 Z1.536 is approached in N310.

N320 changes only the Z position to the value -8.464 as a result of the MOVIT command. As only the table can be rotated, the tool orientation remains unchanged parallel to the machine Z direction, even if the Z direction of the active frame is rotated by 45 degrees.

N330 deletes the basic or system frame; thus the frame definition from N280 is undone.

In N340, TCOFR specifies that the toolholder with orientation capability is to be aligned according to the active frame. Since a rotation is no longer active in N330 due to the PAROTOF command, the initial state is applied. The frame offset becomes zero.

N350 thus approaches the position X10 X10 Z0 (= Z-13 + tool length). Note: Through the simultaneous programming of both rotary axes A and B the actual position of the toolholder with orientation capability is made to match that used in N340. The position approached by the three linear axes is dependent on this position, however.

In N360, solid angles are used to define a plane whose intersecting lines in the XZ and in the YZ plane each form an angle of +45 degrees or -45 degrees with the X or Y axis. The plane defined in such a way therefore has the following position: the surface normal points towards the solid diagonals.

N370 traverses to the position X20 Y0 Z0 in the new coordinate system. Since the tool is deselected with D0 at the same time, there is no longer an additional offset in Z. Since the new X axis lies in the old XZ plane, this block reaches the machine position X14.142 Y0 Z-14.142.

N380 only traverses on the Y axis in the rotated coordinate system. This leads to a motion of all three machine axes. The machine position is X5.977 Y16.330 Z-22.307.

N390 approaches a point on the new Z axis. Relative to the machine axes this is thus on the solid diagonal. All three axes thus reach the position 11.547.

### 18.15.1.3 Basic tool orientation example

#### Basic orientation in the bisector

A milling tool is defined with length $l_1=10$ whose basic orientation is in the bisector of the XZ plane.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>$\text{TC_DP1}[1,1]=120$</td>
</tr>
<tr>
<td>N20</td>
<td>$\text{TC_DP3}[1,1]=10$</td>
</tr>
<tr>
<td>N30</td>
<td>$\text{TC_DPV}[1,1]=0$</td>
</tr>
<tr>
<td>N40</td>
<td>$\text{TC_DPV3}[1,1]=1$</td>
</tr>
<tr>
<td>N50</td>
<td>$\text{TC_DPV4}[1,1]=0$</td>
</tr>
<tr>
<td>N60</td>
<td>$\text{TC_DPV5}[1,1]=1$</td>
</tr>
</tbody>
</table>
### Description of example:

In N10 to N60, a milling tool is defined with length \( L_1 = 10 \) (N20). The basic orientation is in the bisector of the XZ plane N40 to N60.

In N70, the tool is activated and the zero position is approached. As a result of the tool length, the machine positions \( X_0 \ Y_0 \ Z_{10} \) are thus obtained in this block.

In N80, an incremental traversing motion is performed from 10 into tool direction. The resulting axis positions are thus \( X_{7.071} \ Y_0 \ Z_{17.071} \).

#### 18.15.1.4 Calculation of compensation values on a location-specific and workpiece-specific basis

**Tool with adapter**

A tool with adapter and toolholder with orientation capability is defined in the following program example. In order to simplify the overview, only length \( L_1 \) is different to zero for the additive and insert offsets and for the adapter in case of the tool itself. The offset vectors of the toolholder with orientation capability are all zero.

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>$TC_TP2[1] = &quot;MillingTool&quot; ; Name of identifier</td>
</tr>
<tr>
<td>N20</td>
<td>$TC_TP7[1]=9 ; Location types</td>
</tr>
<tr>
<td>N30</td>
<td>$TC_TP8[1]=2 ; Status: Enabled and not blocked</td>
</tr>
<tr>
<td>N40</td>
<td>; D corr. D=1</td>
</tr>
<tr>
<td>N50</td>
<td>$TC_DP1[1,1]=120 ; Tool type - milling</td>
</tr>
<tr>
<td>N60</td>
<td>$TC_DP3[1,1]= ; tool length compensation vector</td>
</tr>
<tr>
<td>N70</td>
<td>$TC_SCP13[1,1]=0.1 ; Sum offset DL=1</td>
</tr>
<tr>
<td>N80</td>
<td>$TC_ECP13[1,1]=0.01 ; Insert offset DL=1</td>
</tr>
<tr>
<td>N90</td>
<td>$TC_ADPTT[1]=5 ; Adapter transformation</td>
</tr>
<tr>
<td>N100</td>
<td>$TC_ADPT1[1]=0.001 ; Adapter dimension</td>
</tr>
<tr>
<td>N110</td>
<td>; Magazine data</td>
</tr>
<tr>
<td>N120</td>
<td>$TC_MAP1[1]=3 ; Magazine type: Revolver</td>
</tr>
<tr>
<td>N130</td>
<td>$TC_MAP2[1]=&quot;Revolver&quot; ; Magazine name</td>
</tr>
<tr>
<td>N140</td>
<td>$TC_MAP3[1]=17 ; Status of magazine</td>
</tr>
<tr>
<td>N150</td>
<td>$TC_MAP6[1]=1 ; Dimension - line</td>
</tr>
<tr>
<td>N160</td>
<td>$TC_MAP7[1]=2 ; Dimension - column -&gt; 2 positions</td>
</tr>
<tr>
<td>N170</td>
<td>$TC_MPP1[1,1]=1 ; Location type</td>
</tr>
<tr>
<td></td>
<td>$TC_MPP2[1,1]=9 ; Location types</td>
</tr>
</tbody>
</table>
N180 $TC_MPP4[1,1]=2 ; Location state
N190 $TC_MPP7[1,1]=1 ; Bring adapter into position
N200 $TC_MPP6[1,1]=1 ; T number "MillingTool"
N210 $TC_MAP1[9999]=7 ; Magazine type: buffer
N220 $TC_MAP2[9999]="buffer" ; Magazine name
N230 $TC_MAP3[9999]=17 ; Status of magazine
N240 $TC_MAP6[9999]=1 ; Dimension - line
N250 $TC_MAP7[9999]=1 ; Dimension - column -> 1 position
N260 $TC_MPP1[9999.1]=2 ; Location type
N270 $TC_MPP2[9999.1]=9 ; Location types
N280 $TC_MPP4[9999.1]=2 ; Location state
N290 $TC_MPP5[9999.1]=1 ; Spindle no. 1
N300 $TC_MDP2[1,1]=0 ; Distance from spindle to mag. 1
N310 $TC_CARR10[1] = 1 ; Definition of toolholder 1
N320 $TC_CARR14[1] = 45 ; Component of 2nd rotary axis in X direction
N330 $TC_CARR23[1] = "T" ; Angle of rotation of 2nd axis
N340 Stopre
N350 $SC_WEAR_TRANSFORM = 'B101'
N360 T0 D0 DL=0
N370 ROT X30
N380 G90 G1 G17 F10000 X0 Y0 Z0
N390 T="MillingTool" X0 Y0 Z0 TOWSTD ; X 0.000 Y11.110 Z 0.001
N400 T="MillingTool" X0 Y0 Z0 TOWMCS ; X 0.000 Y10.100 Z 1.011
N410 T="MillingTool" X0 Y0 Z0 TOWWCS ; X 0.000 Y 9.595 Z 0.876
N420 TCARR=1 X0 Y0 Z0 ; X 0.000 Y 6.636 Z 8.017
N430 G18 X0 Y0 Z0 ; X10.100 Y-0.504 Z 0.876
N440 m30

Explanations regarding the example above

Starting at block N390, various methods are used to approach position X0 Y0 Z0. The machine positions reached are specified in the blocks in comments. After the program a description is given of how the positions were reached.

N390: The adapter transformation 5 (block N90) transforms length L1 into length L2. Only the actual adapter dimension is not subject to this transformation. The Y value (L2 with G17) results from the sum of the tool length (10), tool wear (1), sum offset (0.1), and insert offset (0.01). The adapter dimension (0.001) is in Z (L1).

N400: In block N350, bits 0 and 2 are enabled in setting data:

SD42935 $SC_WEAR_TRANSFORM (transformations for tool components)

This means that the tool wear and the insert offset are not subject to the adapter transformation because of TOWMCS in block N400. The sum of these two compensations is 1.01. The Z position
is, therefore, increased by this amount and the Y position is reduced by this amount compared with block N390.

**TOWWCS** is active in N410. The sum of the tool wear and the insert offset is thus effective in the active workpiece coordinate system. In block N370, a rotation through 30 degrees is activated around the X axis. The original compensation value of 1.01 in the Z direction thus yields a new Z component of 0.875 ( = 1.01 * cos(30)) and a new Y component of -0.505 ( = 1.01 * sin(30)). This yields the dimension specified in the program comment when added to the sum of the tool length, sum offset and adapter dimension produced in block N390.

In addition, a toolholder with orientation capability is activated in block N420. This executes a rotation through 45 degrees about the X axis (see N310 - N330). Since all offset vectors of the toolholder are zero, there is no additional zero offset. The toolholder with orientation capability acts on the sum of the tool length, sum offset and adapter dimension. The resulting vector component is \( X_0 \ Y_7.141 \ Z 7.142 \). To this, as in block N410, the sum of tool wear and insert offset evaluated in WCS is added.

G18 is activated in N430. The components of the tool length sum, sum offset and adapter dimension are interchanged accordingly. The toolholder with orientation capability continues to act on this new vector (rotation through 45 degrees about X axis). The resulting vector component thereby is \( X10.100 \ Y0.0071 \ Z0.0071 \). The vector formed from tool wear and insert offset \( X_0 \ Y-0.505 \ Z0.875 \) is not affected by the change of plane. The sum of the two vectors yields the dimension specified in the comment in N430.

### 18.15.2 Examples 3-6: SETTCOR function for tool environments

**Example 3**

```plaintext
N10   def real _CORVAL[3]  
N20   $TC_DP1[1,1] = 120  ; Milling tool  
N30   $TC_DP3[1,1] = 10.0  ; Geometry L1  
N40   $TC_DP12[1,1] = 1.0  ; Wear L1  
N50   _CORVAL[0] = 0.333  
N60   t1 d1 g17 g0  
N70   r1 = settcor(_CORVAL, "GW", 0, 2, 2)  
N80   t1 d1 x0 y0 z0  ; --> MCS position X0.000 Y0.000 Z0.333  
N90   M30  

_CORCOMP is 2, therefore, the compensation effective in the Z direction is entered in the geometry component (the old value is overwritten) and the wear value is deleted. The resulting total tool length is thus:

\[ L_1 = 0.333 + 0.0 = 0.333. \]

**Example 4**

```plaintext
N10   def real _CORVAL[3]  
```
W1: Tool offset

18.15 Examples

.example 5

 def real _CORVAL[3]
 N10 $TC_DP1[1,1] = 120 ; Milling tool
 N20 $TC_DP3[1,1] = 10.0 ; Geometry L1
 N30 $TC_DP12[1,1] = 1.0 ; Wear L1
 N40 _CORVAL[0] = 0.333
 N50 t1 d1 g17 g0
 N60 r1 = settcor(_CORVAL, "GW", 0, 3, 2)
 N70 t1 d1 x0 y0 z0 ;==> MCS position X0.000 Y0.000 Z11.333
 N80 M30

_CORCOMP is 3, therefore, the wear value and compensation value are added to the geometry component and the wear component is deleted. The resulting total tool length is thus \( L1 = 11.333 + 0.0 = 11.333 \).

.example 6

 def real _CORVAL[3]
 N10 $TC_DP1[1,1] = 500 ; Turning tool
 N20 $TC_DP3[1,1] = 10.0 ; Geometry L1
 N30 $TC_DP4[1,1] = 15.0 ; Geometry L2
 N40 $TC_DP12[1,1] = 10.0 ; Wear L1
 N50 $TC_DP13[1,1] = 0.0 ; Wear L2
 N60 _CORVAL[0] = 5.0
 N70 rot y 30
 N80 t1 d1 g18 g0
 N90 r1 = settcor(_CORVAL, "GW", 0, 3, 1)
 N100 M30

_CORCOMP is 3, as in the previous example, but the compensation is now effective on the geometry axis with index 0 (X axis). The tool components \( L3 \) are assigned to this geometry axis due to G17 with a milling tool. Calling SETTCOR thus does not affect tool parameters \$TC_DP3 \) and \$TC_DP12. Instead, the compensation value is entered in \$TC_DP5.
The tool is a turning tool. A frame rotation is activated in N80, causing the basic coordinate system (BCS) to be rotated in relation to the workpiece coordinate system (WCS). In the WCS, the compensation value (N70) acts on the geometry axis with index 1, i.e., on the X axis because G18 is active. Since "_CORRMODE = 3", the tool wear in the direction of the X axis of the WCS must become zero once N100 has been executed. The contents of the relevant tool parameters at the end of the program are thus:

$TC_{DP3}[1,1]$ : 21.830 ; Geometry L1
$TC_{DP4}[1,1]$ : 21.830 ; Geometry L2
$TC_{DP12}[1,1]$ : 2.500 ; Wear L1
$TC_{DP13}[1,1]$ : -4.330 ; Wear L2

The total wear including _CORVAL is mapped onto the X' direction in the WCS. This produces point P2. The coordinates of this point (measured in X/Y coordinates) are entered in the geometry component of the tool. The difference vector P2 - P1 remains in the wear. The wear thus no longer has a component in the direction of _CORVAL.

Figure 18-54 Tool length compensation, example 6

If the sample program is continued after N110 with the following instructions:

N120 _CORVAL[0] = 0.0
N130 r1 = settcor(_CORVAL, "GW", 0, 3, 0)
N140 t1 d1 x0 y0 z0 ; ==> MCS position X24.330 Y0.000 Z17.500
The remaining wear is included completely in the geometry because the compensation is now effective in the Z' axis (parameter GEOAX is 0). Since the new compensation value is 0, the total tool length and thus the position approached in N140 may not change. If _CORVAL were not equal to 0 in N120, a new total tool length and thus a new position in N140 would result, however, the wear component of the tool length would always be zero, i.e., the total tool length is subsequently always contained in the geometry component of the tool.

The same result as that achieved by calling the SETTCOR function with the _CORCOMP = 0 parameter twice can also be reached by calling _CORRCOMP = 1 (vectorial compensation) just once:

```plaintext
N10  def real _CORVAL[3]
N20  $TC_DP1[1,1] = 500 ; Turning tool
N30  $TC_DP3[1,1] = 10.0 ; Geometry L1
N40  $TC_DP4[1,1] = 15.0 ; Geometry L2
N50  $TC_DP12[1,1] = 10.0 ; Wear L1
N60  $TC_DP13[1,1] = 0.0 ; Wear L2
N70  _CORVAL[0] = 0.0
N71  _CORVAL[1] = 5.0
N72  _CORVAL[2] = 0.0
N80  rot y 30
N90  t1 d1 g18 g0
N100 r1 = settcor(_CORVAL, "GW", 1, 3, 1)
N110 t1 d1 x0 y0 z0 ; ==> MCS position X24.330 Y0.000 Z17.500
N120 M30
```

In this case, all wear components of the tool are set to zero immediately after the first call of SETTCOR in N100.

**Example 7**

```plaintext
N10  def real _CORVAL[3]
N20  $TC_DP1[1,1] = 500 ; Turning tool
N30  $TC_DP3[1,1] = 10.0 ; Geometry L1
N40  $TC_DP4[1,1] = 15.0 ; Geometry L2
N50  $TC_DP12[1,1] = 10.0 ; Wear L1
N60  $TC_DP13[1,1] = 0.0 ; Wear L2
N70  _CORVAL[0] = 5.0
N80  rot y 30
N90  t1 d1 g18 g0
N100 r1 = settcor(_CORVAL, "GW", 3, 3)
N110 t1 d1 x0 y0 z0 ; ==> MCS position X25.000 Y0.000 Z15.500
N120 M30
```
As opposed to example 6, parameter \_CORCOMP = 3, and so the \_GEOAX parameter can be omitted. The value contained in \_CORVAL[0] now acts immediately on the tool length component L1, the rotation in N80 has no effect on the result, the wear components in $TC\_DP12 are included in the geometry component together with \_CORVAL[0], with the result that the total tool length is stored in the geometry component of the tool, due to $TC\_DP13, after the first SETTCOR call in N100.

Example 8

\begin{verbatim}
N10   def real _CORVAL[3]
N20   $TC\_DP1[1,1] = 500 ; Turning tool
N30   $TC\_DP3[1,1] = 10.0 ; Geometry L1
N40   $TC\_DP4[1,1] = 15.0 ; Geometry L2
N50   $TC\_DP5[1,1] = 20.0 ; Geometry L3
N60   $TC\_DP12[1,1] = 10.0 ; Wear L1
N70   $TC\_DP13[1,1] = 0.0 ; Wear L2
N80   $TC\_DP14[1,1] = 0.0 ; Wear L3
N90   $SC\_WEAR\_SIGN = TRUE
N100  _CORVAL[0] = 10.0
N110  _CORVAL[1] = 15.0
N120  _CORVAL[2] = 5.0
N130  rot y 30
N140  t1 d1 g18 g0
N150  r1 = settcor(_CORVAL, "W", 1, 1)
N160  t1 d1 x0 y0 z0 ; --> MCS position X7.990 Y25.000 Z31.160
\end{verbatim}

In N90 the setting data is enabled:

SD42930 $SC\_WEAR\_SIGN (sign of wear)

i.e. the wear must be valued with a negative sign.

The compensation is vectorial (_CORCOMP = 1), and the compensation vector must be added to the wear (_CORMODE = 1). The geometric conditions in the Z/X plane are shown in the figure below:
The geometry component of the tool remains unchanged due to _CORMODE = 1. The compensation vector defined in the WCS (rotation about y axis) must be included in the wear component such that the total tool length in the figure refers to point P₂. Therefore, the resulting wear component of the tool is given in relation to the distance between points P₁ and P₂.

However, since the wear is evaluated negatively, due to setting data SD42930, the compensation determined in this way has to be entered in the compensation memory with a negative sign. The contents of the relevant tool parameters at the end of the program are thus:

| $TC_DP3[1,1]   | 10.000 | Geometry L1 (unchanged) |
| $TC_DP4[1,1]   | 15.000 | Geometry L2 (unchanged) |
| $TC_DP5[1,1]   | 10.000 | Geometry L3 (unchanged) |
| $TC_DP12[1,1]  | 2.010  | Wear L₁ \(-10 -15\times\cos(30) + 10\times\sin(30)\) |
| $TC_DP13[1,1]  | -16.160 | Wear L₂ \(-15\times\sin(30) - 10\times\cos(30)\) |
| $TC_DP14[1,1]  | -5.000  | Wear L₃ |

The effect of setting data SD42930 on the L₃ component in the Y direction can be recognized without the additional complication caused by the frame rotation.

**Example 9:**

2 (tool length must be valued in the diameter axis with the factor 0.5) is the value of machine data:

MD20360 $MC_TOOL_PARAMETER_DEF_MASK (definition of tool parameters).
X is diameter axis:

```plaintext
N10  def real _LEN[11]
N20  def real _CORVAL[3]
N30  $TC_DP1[1,1]= 500
N40  $TC_DP2[1,1]= 2
N50  $TC_DP3[1,1]= 3.
N60  $TC_DP4[1,1]= 4.
N70  $TC_DP5[1,1]= 5.
N80  _CORVAL[0] = 1.
N110 t1 d1 g18 g0 x0 y0 z0 ; ==> MCS position X1.5 Y5 Z4
N120 r1 = settcor(_CORVAL, "g", 1, 1) ; ==> MCS position X2.5 Y6 Z5
N130 t1 d1 x0 y0 z0
N140 r3 = $TC_DP3[1,1] ; = - 5. = (3.000 + 2. * 1.000)
N150 r4 = $TC_DP4[1,1] ; = 5. = (4.000 + 1.000)
N160 r5 = $TC_DP5[1.1] ; = 6. = (5.000 + 1.000)
N170 m30
```

The compensation of the tool length is to be 1 mm in each axis (N80 to N100). 1 mm is thus added to the original length in lengths L2 and L3.

Twice the compensation value (2 mm) is added to the original tool length in L1, in order to change the total length by 1 mm as required. If the positions approached in blocks N110 and N130 are compared, it can be seen that each axis position has changed by 1 mm.

### 18.16 Data lists

#### 18.16.1 Machine data

##### 18.16.1.1 NC-specific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN_</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>18082</td>
<td>MM_NUM_TOOL</td>
<td>Number of tools that the NCK can manage (SRAM)</td>
</tr>
<tr>
<td>18088</td>
<td>MM_NUM_TOOL_CARRIER</td>
<td>Maximum number of the defined toolholders</td>
</tr>
<tr>
<td>18094</td>
<td>MM_NUM_CC_TDA_PARAM</td>
<td>Number of tool data (SRAM)</td>
</tr>
<tr>
<td>18096</td>
<td>MM_NUM_CC_TOA_PARAM</td>
<td>Number of data, per tool cutting edge for compile cycles (SRAM)</td>
</tr>
<tr>
<td>18100</td>
<td>MM_NUM_CUTTING_EDGES_IN_TOA</td>
<td>Tool offsets in the TOA area (SRAM)</td>
</tr>
<tr>
<td>18102</td>
<td>MM_TYPE_OF_CUTTING_EDGE</td>
<td>Type of D number programming (SRAM)</td>
</tr>
</tbody>
</table>
### 18.16 Data lists

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MN__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>18105</td>
<td>MM_MAX_CUTTING_EDGE_NO</td>
<td>Maximum value of D number</td>
</tr>
<tr>
<td>18106</td>
<td>MM_MAX_CUTTING_EDGE_PERTOOL</td>
<td>Maximum number of D numbers per tool</td>
</tr>
<tr>
<td>18108</td>
<td>MM_NUM_SUMCORR</td>
<td>Number of all sum offsets in NCK</td>
</tr>
<tr>
<td>18110</td>
<td>MM_MAX_SUMCORR_PER_CUTTEDGE</td>
<td>Number of additive offsets per cutting edge</td>
</tr>
<tr>
<td>18112</td>
<td>MM_KIND_OF_SUMCORR</td>
<td>Properties of additive offsets in the TO area (SRAM)</td>
</tr>
<tr>
<td>18114</td>
<td>MM_ENABLE_TOOL_ORIENT</td>
<td>Assign orientation to cutting edges</td>
</tr>
<tr>
<td>18116</td>
<td>MM_NUM_TOOL_ENV</td>
<td>Tool environments in the TOA area (SRAM)</td>
</tr>
</tbody>
</table>

### 18.16.1.2 Channelspecific machine data

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MC__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20096</td>
<td>T_M_ADDRESS_EXT_IS_SPINO</td>
<td>Meaning of the address extension with T, M tool change</td>
</tr>
<tr>
<td>20110</td>
<td>RESET_MODE_MASK</td>
<td>Definition of initial control setting after RESET/part program end</td>
</tr>
<tr>
<td>20120</td>
<td>TOOL_RESET_VALUE</td>
<td>Tool length compensation at power-up (Reset/TP end)</td>
</tr>
<tr>
<td>20121</td>
<td>TOOL_PRESSEL_RESET_VALUE</td>
<td>Preselected tool on RESET</td>
</tr>
<tr>
<td>20125</td>
<td>CUTMOD_ERR</td>
<td>Troubleshooting for the CUTMOD function</td>
</tr>
<tr>
<td>20126</td>
<td>TOOL_CARRIER_RESET_VALUE</td>
<td>Active toolholder on RESET</td>
</tr>
<tr>
<td>20127</td>
<td>CUTMOD_INIT</td>
<td>Initialize CUTMOD for POWER ON</td>
</tr>
<tr>
<td>20130</td>
<td>CUTTING_EDGE_RESET_VALUE</td>
<td>Tool cutting edge length compensation at power-up (Reset/TP end)</td>
</tr>
<tr>
<td>20132</td>
<td>SUMCORR_RESET_VALUE</td>
<td>Additive offset effective at RESET</td>
</tr>
<tr>
<td>20140</td>
<td>TRAFO_RESET_VALUE</td>
<td>Transformation data record at power up (Reset/TP end)</td>
</tr>
<tr>
<td>20180</td>
<td>TOCARR_ROT_ANGLE_INCR[i]</td>
<td>Rotary axis increment of the toolholder with orientation capability</td>
</tr>
<tr>
<td>20182</td>
<td>TOCARR_ROT_ANGLE_OFFSET[i]</td>
<td>Rotary axis offset of toolholder with orientation capability</td>
</tr>
<tr>
<td>20184</td>
<td>TOCARR_BASE_FRAME_NUMBER</td>
<td>Number of the basic frames to accept the table offset</td>
</tr>
<tr>
<td>20188</td>
<td>TOCARR_FINE_LIM_LIN</td>
<td>Limit linear fine offset TCARR</td>
</tr>
<tr>
<td>20190</td>
<td>TOCARR_FINE_LIM_ROT</td>
<td>Limit of the rotary fine offset TCARR</td>
</tr>
<tr>
<td>20202</td>
<td>WAB_MAXNUM_DUMMY_BLOCKS</td>
<td>Maximum number of blocks with no traversing motions with SAR</td>
</tr>
<tr>
<td>20204</td>
<td>WAB_CLEARANCE_TOLERANCE</td>
<td>Direction reversal for WAB</td>
</tr>
<tr>
<td>20210</td>
<td>CUTCOM_CORNER_LIMIT</td>
<td>Max. angle for intersection calculation with tool radius compensation</td>
</tr>
<tr>
<td>20220</td>
<td>CUTCOM_MAX_DISC</td>
<td>Maximum value for DISC</td>
</tr>
<tr>
<td>20230</td>
<td>CUTCOM_CURVE_INSERT_LIMIT</td>
<td>Maximum value for intersection calculation with TRC</td>
</tr>
<tr>
<td>20240</td>
<td>CUTCOM_MAXNUM_CHECK_BLOCKS</td>
<td>Blocks for predictive contour calculation with tool radius compensation</td>
</tr>
<tr>
<td>20250</td>
<td>CUTCOM_MAXNUMDummy_BLOCKS</td>
<td>Maximum number of blocks without traversing motion for TRC</td>
</tr>
<tr>
<td>20252</td>
<td>CUTCOM_MAXNUM_SUPPR_BLOCKS</td>
<td>Maximum number of blocks with compensation suppression</td>
</tr>
</tbody>
</table>
### 18.16 Data lists

**18.16.1.3 Axis/spindlespecific machine data**

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $MA__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>32750</td>
<td>TEMP_COMP_TYPE</td>
<td>Temperature compensation type</td>
</tr>
</tbody>
</table>

**18.16.2 Setting data**

**18.16.2.1 Channelspecific setting data**

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $SC__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>42442</td>
<td>TOOL_OFFSET_INCR_PROG</td>
<td>Tool length compensation</td>
</tr>
<tr>
<td>42470</td>
<td>CRIT_SPLINE_ANGLE</td>
<td>Core limit angle, for compressor</td>
</tr>
<tr>
<td>42480</td>
<td>STOP_CUTCOM_STOPRE</td>
<td>Alarm response for tool radius compensation and preprocessing stop</td>
</tr>
<tr>
<td>42494</td>
<td>CUTCOM_ACT_DEACT_CTRL</td>
<td>Approach and retraction behavior for tool radius compensation</td>
</tr>
<tr>
<td>42496</td>
<td>CUTCOM_CLSDT_CONT</td>
<td>Behavior of the tool radius compensation for closed contour</td>
</tr>
<tr>
<td>42900</td>
<td>MIRROR_TOOL_LENGTH</td>
<td>Sign change, tool lengths when mirroring</td>
</tr>
<tr>
<td>42910</td>
<td>MIRROR_TOOL_WEAR</td>
<td>Sign change, tool wear when mirroring</td>
</tr>
<tr>
<td>42920</td>
<td>WEAR_SIGN_CUTPOS</td>
<td>Sign of wear for tools with cutting edge position</td>
</tr>
<tr>
<td>42930</td>
<td>WEAR_SIGN</td>
<td>Sign of the wear</td>
</tr>
</tbody>
</table>
### 18.16 Data lists

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier: $SC__</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>42935</td>
<td>WEAR_TRANSFORM</td>
<td>Transformations for tool components</td>
</tr>
<tr>
<td>42940</td>
<td>TOOL_LENGTH_CONST</td>
<td>Change of tool length components for change of plane</td>
</tr>
<tr>
<td>42950</td>
<td>TOOL_LENGTH_TYPE</td>
<td>Assignment of the tool length offset independent of tool type</td>
</tr>
<tr>
<td>42960</td>
<td>TOOL_TEMP_COMP</td>
<td>Temperature compensation value in relation to tool</td>
</tr>
<tr>
<td>42974</td>
<td>TCARR_FINE_CORRECTION</td>
<td>Fine offset TCARR on/off</td>
</tr>
</tbody>
</table>

### 18.16.3 Signals

#### 18.16.3.1 Signals from channel

<table>
<thead>
<tr>
<th>Signal name</th>
<th>SINUMERIK 840D sl</th>
<th>SINUMERIK 828D</th>
</tr>
</thead>
<tbody>
<tr>
<td>T function 1 change</td>
<td>DB21, ... .DBX61.0</td>
<td>-</td>
</tr>
<tr>
<td>D function 1 change</td>
<td>DB21, ... .DBX62.0</td>
<td>-</td>
</tr>
<tr>
<td>T function 1</td>
<td>DB21, ... .DBB116-119</td>
<td>DB250x.DBD2000</td>
</tr>
<tr>
<td>D function 1</td>
<td>DB21, ... .DBB128-129</td>
<td>DB250x.DBD5000</td>
</tr>
<tr>
<td>Active G function of group 7</td>
<td>DB21, ... .DBB214</td>
<td>DB350x.DBB6</td>
</tr>
<tr>
<td>Active G function of group 16</td>
<td>DB21, ... .DBB223</td>
<td>DB350x.DBB15</td>
</tr>
<tr>
<td>Active G function of group 17</td>
<td>DB21, ... .DBB224</td>
<td>DB350x.DBB16</td>
</tr>
<tr>
<td>Active G function of group 18</td>
<td>DB21, ... .DBB225</td>
<td>DB350x.DBB17</td>
</tr>
<tr>
<td>Active G function of group 23</td>
<td>DB21, ... .DBB230</td>
<td>DB350x.DBB22</td>
</tr>
</tbody>
</table>
19.1 Various interface signals and functions (A2)

19.1.1 Signals from PLC to NC (DB10)

<table>
<thead>
<tr>
<th>DB10</th>
<th>Key-operated switch positions 0 to 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX56.4 - DBX56.7</td>
<td>Signal(s) updated: Cyclic</td>
</tr>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclic</td>
</tr>
<tr>
<td>Significance of signal</td>
<td>Depending on the key-operated switch position, access to certain elements in the NCK can be enabled or disabled.</td>
</tr>
<tr>
<td></td>
<td>• Key-operated switch position 0 represents the lowest access rights</td>
</tr>
<tr>
<td></td>
<td>• Key-operated switch position 3 represents the highest access rights</td>
</tr>
<tr>
<td></td>
<td>The NC/PLC interface signals of key-operated switch positions 1 to 3 can either be directly specified from the key-operated switch on the machine control panel or from the PLC user program.</td>
</tr>
<tr>
<td></td>
<td>It is only permissible to set one bit. If several bits are set simultaneously, the control internally activates switch position 3.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key-operated switch position</th>
<th>DBX56.7</th>
<th>DBX56.6</th>
<th>DBX56.5</th>
<th>DBX56.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Corresponding to ... Machine data for protection levels: MD11612, MD51044 - MD51064, MD51070 - MD51073, MD51199 - MD51211, MD51215 - MD51225, MD51235
Disabling using a password

19.1.2 Selection/Status signals from HMI to PLC (DB10)

<table>
<thead>
<tr>
<th>DB10</th>
<th>Remote diagnosis active</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX103.0</td>
<td>Remote diagnosis active</td>
</tr>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>Remote diagnosis (optional) is active, i.e. the control is operated via an external PC.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Remote diagnosis is not active.</td>
</tr>
</tbody>
</table>
DBX103.5  AT box ready

- Edge evaluation: No
- Signal(s) updated: Cyclically
- Signal state 1 or edge change 0 → 1: The AT box for expansion modules is ready.
- Signal state 0 or edge change 1 → 0: The AT box is not ready. An expansion module conforming to the AT specification has either no functionality or restricted functionality.

DBX103.6  HMI temperature limit

- Edge evaluation: No
- Signal(s) updated: Cyclically
- Signal state 1 or edge change 0 → 1: The environment conditions of the limit value are in the permitted tolerance range of 5 to 55°C.
- Signal state 0 or edge change 1 → 0: The temperature range was either exceeded, or the temperature fell below the limit. The temperature monitor has responded and the PCU is disabled.

DBX103.7  HMI battery alarm

- Edge evaluation: No
- Signal(s) updated: Cyclically
- Signal state 1 or edge change 0 → 1: The battery monitor has responded. Power failure can cause the loss of recently changed data and a correct device configuration. An appropriate alarm is issued. The buffer battery should be checked. An insufficient battery voltage also affects the current time on the user interface.
- Signal state 0 or edge change 1 → 0: No HMI battery alarm is present.
- Additional references: Operator Components Manual (PCU)

19.1.3 Signals from the NC to the PLC (DB10)

DBX104.7  NCK CPU ready

- Edge evaluation: No
- Signal(s) updated: Cyclically
- Signal state 1: The NCK CPU is ready and registers itself cyclically with the PLC. After a correct initial start and the first complete OB1 cycle (initial setting cycle) the PLC and NCK continuously exchange sign-of-life signals. The PLC basic program sets the interface signal "NCK CPU Ready" to 1.
### DB10 DBX104.7 | NCK CPU ready
---|---
**Signal state 0** | The NCK CPU is not ready.  
If a sign-of-life is not received from the NCK, the PLC/NCK interface is neutralized by the PLC basic program and the interface signal "NCK CPU Ready" is set to 0.  
The following measures are introduced by the PLC basic program:  
- Status signals from NCK to PLC (user interface) are deleted (cleared)  
- Change signals for help functions are deleted  
- Cyclic processing of the user interface PLC to NCK is terminated.  
**Additional references** | • Diagnostics Manual  
• Function Manual Basic Functions; Basic PLC Program

### DB10 DBX108.3 | SINUMERIK Operate at OPI ready
---|---
**Edge evaluation:** No  
**Signal(s) updated:** Cyclically  
**Signal state 1** | SINUMERIK Operate is ready and cyclically logs on with the NCK.  
**Signal state 0** | SINUMERIK Operate is not ready.  
**Additional references** | Diagnostics Manual

### DB10 DBX108.5 | Drives in cyclic operation
---|---
**Edge evaluation:** No  
**Signal(s) updated:** Cyclically  
**Signal state 1** | For all machine axes of the NC, the corresponding drives are in the cyclic operation, i.e. they cyclically exchange PROFldrive telegrams with the NC.  
**Signal state 0** | For at least one machine axis of the NC, the corresponding drive is not in cyclic operation, i.e. it is not cyclically exchanging PROFldrive telegrams with the NC.

### DB10 DBX108.6 | Drive ready
---|---
**Edge evaluation:** No  
**Signal(s) updated:** Cyclically  
**Signal state 1** | For all machine axes of the NC, the corresponding drives are ready:  
DB31, ... DBX93.5 == 1 (DRIVE ready)  
**Signal state 0** | For at least one machine axis of the NC, the corresponding drive is not ready:  
DB31, ... DBX93.5 == 0 (DRIVE ready)  
**Corresponding to ...** | DB31, ... DBX93.5 (DRIVE ready)
### DB10 DBX108.7 NC ready

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal state 1</strong></td>
<td>The control system is ready.</td>
</tr>
<tr>
<td></td>
<td>This interface signal is an image of the relay contact &quot;NC Ready&quot;.</td>
</tr>
<tr>
<td></td>
<td>This signal is set if:</td>
</tr>
<tr>
<td></td>
<td>● Relay contact &quot;NC Ready&quot; is closed</td>
</tr>
<tr>
<td></td>
<td>● All the voltages in the controller have been established</td>
</tr>
<tr>
<td></td>
<td>● The controller is in the cyclic mode</td>
</tr>
<tr>
<td><strong>Signal state 0</strong></td>
<td>The controller is not ready. The relay contact &quot;NC Ready&quot; is open.</td>
</tr>
<tr>
<td></td>
<td>The following faults will cause NC Ready to be canceled:</td>
</tr>
<tr>
<td></td>
<td>● Undervoltage and overvoltage monitoring function has responded</td>
</tr>
<tr>
<td></td>
<td>● Individual components are not ready (NCK CPU Ready)</td>
</tr>
<tr>
<td></td>
<td>● NCK CPU watchdog</td>
</tr>
<tr>
<td></td>
<td>If the signal &quot;NC Ready&quot; goes to 0 the following measures are introduced by the controller if they are still possible:</td>
</tr>
<tr>
<td></td>
<td>● The controller enable signals are withdrawn (this stops the drives)</td>
</tr>
<tr>
<td></td>
<td>● The following measures are introduced by the PLC basic program:</td>
</tr>
<tr>
<td></td>
<td>– Status signals from NCK to PLC (user interface) are deleted (cleared)</td>
</tr>
<tr>
<td></td>
<td>– Change signals for help functions are deleted</td>
</tr>
<tr>
<td></td>
<td>– Cyclic processing of the PLC to NCK user interface is terminated</td>
</tr>
<tr>
<td></td>
<td>For further information see References!</td>
</tr>
<tr>
<td></td>
<td>The controller is not ready again until after POWER ON.</td>
</tr>
</tbody>
</table>

**Corresponding to**... Relay contact "NC Ready"

**Additional references**
- Diagnostics Manual
- Function Manual Basic Functions; Basic PLC Program

### DB10 DBX109.0 NCK alarm is active

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal state 1</strong></td>
<td>At least one NCK alarm is present.</td>
</tr>
<tr>
<td></td>
<td>This is a group signal for the interface signals of all existing channels:</td>
</tr>
<tr>
<td></td>
<td>DB21, ... DBX36.6 (channel-specific NCK alarm is active)</td>
</tr>
<tr>
<td><strong>Signal state 0</strong></td>
<td>No NCK alarm is active.</td>
</tr>
<tr>
<td><strong>Corresponding to</strong>... DB21, ... DBX36.6 (channel-specific NCK alarm is active)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB21, ... DBX36.7 (NCK alarm with processing stop present)</td>
</tr>
</tbody>
</table>

**Additional references** Diagnostics Manual
**DB10 DBX109.5**  
**NCU heat sink temperature alarm**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>In the NCU, the limit values of the heat sink temperature was exceeded. The heat sink temperature has been activated and a steady operation of the operator panel is no longer guaranteed.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>The heat sink monitoring function of the NCU has not responded.</td>
</tr>
</tbody>
</table>

**DB10 DBX109.6**  
**Air temperature alarm**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
</table>
| Signal state 1      | The ambient temperature or fan monitoring function has responded. This may be due to the following causes:  
  - The temperature monitoring has identified an ambient temperature that is too high (approx. 60 °C). Alarm 2110 "NCK temperature alarm" is output.  
  - The speed monitoring of the 24 VDC fan used to cool the module has responded. Alarm 2120 "NCK fan alarm" is output.  
  Measures: Replace the fan and/or ensure that additional ventilation is provided.  
  When a temperature or fan error responds, a relay contact (terminal 5.1, 5.2 or 5.1, 5.3) in the infeed/regenerative feedback unit is activated which can be evaluated by the customer. |
| Signal state 0      | Neither the temperature monitoring nor the fan monitoring has responded. |

**Application example(s)**  
Appropriate measures can be initiated by the PLC user program if the temperature or fan monitoring is activated.

**Corresponding to ...**  
When a temperature or fan error responds, a relay contact (terminal 5.1, 5.2 or 5.1, 5.3) in the infeed/regenerative feedback unit is activated. This can be evaluated.

**Additional references**  
Diagnostics Manual

**DB10 DBX109.7**  
**NCK battery alarm**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
</table>
| Signal state 1      | The NCK battery voltage monitoring function has responded. This may be due to the following causes:  
  - The battery voltage is within the pre-warning limit range (approx. 2.7 to 2.9 V). Alarm 2100 "NCK battery warning threshold reached" has been triggered.  
    Refer to References for effects and measure.  
  - The battery voltage is below the pre-warning limit range (≤ 2.6 V). Alarm 2101 "NCK battery alarm" is signaled in cyclic operation. Effects: A supply voltage failure - e.g. when the controller is switched off - would result in the loss of battery-buffered data (e.g. part program memory, variables, machine data ...).  
    Measure: Refer to additional References.  
  - When the controller powered-up, it was identified that the battery voltage was below the pre-warning limit range (≤ 2.6 V). Alarm 2102 "NCK battery alarm" is output; NC ready and mode group ready are not issued. Effects: Some of the battery-buffered data may already have been lost!  
    Measure: Refer to additional References. |

**Additional references**  
Diagnostics Manual
### 19.1 NCK battery alarm

<table>
<thead>
<tr>
<th>DB10</th>
<th>DBX109.7</th>
<th>NCK battery alarm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 0</td>
<td>The battery voltage is above the lower limit value (normal condition).</td>
<td></td>
</tr>
<tr>
<td>Special cases, errors, ...</td>
<td>The NCK batteries should only be replaced while the NC is switched on to avoid data loss because of no memory backup.</td>
<td></td>
</tr>
</tbody>
</table>
| Additional references | ● Diagnostics Manual  
● Equipment Manual NCU |

### 19.1.4 Signals to Operator Panel (DB19)

<table>
<thead>
<tr>
<th>DB19</th>
<th>DBX0.0</th>
<th>Screen bright</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
<td></td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>The screen blanking is disabled.</td>
<td></td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>The screen blanking remains in effect.</td>
<td></td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB19 DBX0.1 (darken screen)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB19</th>
<th>DBX0.1</th>
<th>Darken screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
<td></td>
</tr>
</tbody>
</table>
| Signal state 1 or edge change 0 → 1 | The screen is darkened by the PLC user program.  
The automatic screen brightening/darkening is therefore ineffective:  
i.e. the screen does not brighten up automatically on actuating the keyboard. |
| Signal state 0 or edge change 1 → 0 | The screen is controlled by the PLC user program "bright".  
In this signal state, the screen brightening/darkening setting can be derived by the control via the keyboard automatically.  
The screen is darkened if no key is pressed for a period defined via the following machine data on the keyboard:  
MD9006 $MM_DISPLAY_BLACK_TIME (time to darken the screen)  
The screen is brightened the next time a key on the operator panel front is pressed. |
| Application example(s) | Screen saver |
| Special cases, errors, .... | Notice:  
The keyboard of the operator panel front continues to be effective if the interface signal:  
DB19 DBX0.1 (darken screen) = 1  
We therefore recommend that this is disabled using the interface signal:  
DB19 DBX0.2 (key disable) |
| Corresponding to ... | DB19 DBX0.2 (key disable)  
MD9006 $MM_DISPLAY_BLACK_TIME (time to darken the screen) |
### DB19 DBX0.2 – Key disable

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>The keyboard is disabled for the user.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>The keyboard is enabled for the user.</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>If the screen is darkened with the interface signal: DB19 DBX0.1 (darken screen), the keyboard should be actuated simultaneously with the interface signal: DB19 DBX0.2 (key disable) to avoid an unintended operation.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB19 DBX0.1 (darken screen)</td>
</tr>
</tbody>
</table>

### DB19 DBX0.3 – Clear cancel alarms

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>Clear error key on the machine control panel is pressed. All cancel alarms of the NCKs and the control panel are then acknowledged. The PLC application acknowledges the PLC alarms itself. Power On and Reset alarms remain active on the NCK until the cause of the error has been removed.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Clear error key on the machine control panel is not pressed.</td>
</tr>
<tr>
<td>Functionality</td>
<td>Only valid for HMI Advanced.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB19 DBX20.3 (cancel alarm cleared)</td>
</tr>
</tbody>
</table>

### DB19 DBX0.4 – Clear recall alarms

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>Clear error key on the machine control panel is pressed. All cancel alarms of the NCKs and the control panel are then acknowledged. The PLC application acknowledges the PLC alarms itself. POWER On and Reset alarms remain active on the NCK until the cause of the error has been removed.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Clear error key on the machine control panel is not pressed.</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Applies to HMI Advanced only</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB19 DBX20.4 (recall alarm cleared)</td>
</tr>
</tbody>
</table>
The PLC selects the display of actual values in the workpiece coordinate system (WCS). This means that when the machine area is selected, the WCS display is activated; i.e. the machine and the supplementary axes as well as their actual positions and distances-to-go are displayed in the WCS in the "Position" window.

The interface signal is only evaluated when it enters the basic machine screen; this means that the operator, within the machine area, can toggle as required between the particular coordinate systems using the softkeys "actual values MCS" and "actual values WCS".

This means that when the machine area is selected the coordinate system previously selected (WCS or MCS) is reactivated and displayed.

Using the interface signal: DB19, DBX0.7 (actual value in WCS) = 1 each time that the machine area is re-selected, the workpiece coordinate system display frequently required by the operator (WCS), is selected.

Corresponding to ...

DB19 DBX20.7 (changeover MCS/WCS)

Additional references

Operation Guide HMI (corresponding to the used software)

<table>
<thead>
<tr>
<th>DB19 DBX0.7</th>
<th>Actual values in WCS, 0 = MCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal state 1 or edge change 0 → 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>The PLC selects the display of actual values in the workpiece coordinate system (WCS). This means that when the machine area is selected, the WCS display is activated; i.e. the machine and the supplementary axes as well as their actual positions and distances-to-go are displayed in the WCS in the &quot;Position&quot; window. The interface signal is only evaluated when it enters the basic machine screen; this means that the operator, within the machine area, can toggle as required between the particular coordinate systems using the softkeys &quot;actual values MCS&quot; and &quot;actual values WCS&quot;.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal state 0 or edge change 1 → 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>This means that when the machine area is selected the coordinate system previously selected (WCS or MCS) is reactivated and displayed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the interface signal: DB19, DBX0.7 (actual value in WCS) = 1 each time that the machine area is re-selected, the workpiece coordinate system display frequently required by the operator (WCS), is selected.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corresponding to ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB19 DBX20.7 (changeover MCS/WCS)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional references</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation Guide HMI (corresponding to the used software)</td>
</tr>
</tbody>
</table>
### DB19

#### DBX13.5

<table>
<thead>
<tr>
<th>Signal state 0 or edge change 1 → 0</th>
<th>Unload inactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application example(s)</td>
<td>A file transfer can be initiated using the hard disk.</td>
</tr>
</tbody>
</table>

#### DBX13.6

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 0 or edge change 0 → 1</td>
<td>Load active</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Load inactive</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>A file transfer can be initiated using the hard disk.</td>
</tr>
</tbody>
</table>

#### DBX13.7

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 0 or edge change 0 → 1</td>
<td>Selection active</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Selection inactive</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>A file transfer can be initiated using the hard disk.</td>
</tr>
</tbody>
</table>

#### DBX14.0 - DBX14.6

<table>
<thead>
<tr>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>This byte for control of the RS-232-C describes the PLC index for the standard control file that specifies the axis, channel or TO number.</td>
</tr>
</tbody>
</table>

Application example(s)
- DB19 DBX14.7=0 → Act. FS: PLC index that specifies axis, channel or TO No.
- DB19 DBX14.7=1 → Pas. FS: PLC index for the user control file

#### DBX14.7

<table>
<thead>
<tr>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active or passive file system</td>
</tr>
<tr>
<td>Edge evaluation: No</td>
</tr>
<tr>
<td>Signal state 0 or edge change 0 → 1</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
</tr>
</tbody>
</table>
**DB19**

<table>
<thead>
<tr>
<th>DBB15</th>
<th>PLC line offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Description</td>
<td>This byte to control the RS-232-C interface defines the line of the standard or user control file in which the control file to be transferred is specified.</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Dependent on:</td>
</tr>
<tr>
<td></td>
<td>DB19 DBX14.7=0 → Act. FS: PLC line offset in a standard control file</td>
</tr>
<tr>
<td></td>
<td>DB19 DBX14.7=1 → Pas. FS: PLC line offset in a user control file</td>
</tr>
</tbody>
</table>

---

**DB19**

<table>
<thead>
<tr>
<th>DBX16.0 - DBX16.6</th>
<th>PLC index for the user control file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Description</td>
<td>This byte for controlling file transfer via hard disk defines the index for the control file (job list).</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Dependent on:</td>
</tr>
<tr>
<td></td>
<td>DB19 DBX14.7=0 → Act. FS: PLC index for the standard control file</td>
</tr>
<tr>
<td></td>
<td>DB19 DBX14.7=1 → Pas. FS: PLC index for the user control file</td>
</tr>
</tbody>
</table>

---

**DB19**

<table>
<thead>
<tr>
<th>DBX16.7</th>
<th>Active or passive file system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>Passive file system</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Active file system</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>with HMI Advanced always 1</td>
</tr>
</tbody>
</table>

---

**DB19**

<table>
<thead>
<tr>
<th>DBB17</th>
<th>PLC line offset in the user control file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Description</td>
<td>This byte for controlling file transfer via the hard disk defines the line of the user control file in which the control file to be transferred is located.</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Dependent on:</td>
</tr>
<tr>
<td></td>
<td>DB19 DBX14.7=0 → Act. FS: PLC line offset in a standard control file</td>
</tr>
<tr>
<td></td>
<td>DB19 DBX14.7=1 → Pas. FS: PLC line offset in a user control file</td>
</tr>
</tbody>
</table>

---

**DB19**

<table>
<thead>
<tr>
<th>DBX44.0</th>
<th>Mode change disable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>Mode change disable active</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Mode change disable active</td>
</tr>
</tbody>
</table>
### 19.1.5 Signals from operator control panel (DB19)

<table>
<thead>
<tr>
<th>DB19</th>
<th>DBX20.1</th>
<th>Screen dark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
<td></td>
</tr>
<tr>
<td>Signal state 1</td>
<td>The screen is darkened.</td>
<td></td>
</tr>
<tr>
<td>Signal state 0</td>
<td>The screen is not darkened.</td>
<td></td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Using this IS, the PLC can identify whether the screen was switched dark using the interface signal: DB19, DBX0.1 (darken screen) or using the machine data: MD9006 $MM_DISPLAY_BLACK_TIME (screen blanking time) .</td>
<td></td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>MD9006 $MM_DISPLAY_BLACK_TIME (screen blanking time)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB19</th>
<th>DBX20.3</th>
<th>Cancel alarm deleted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
<td></td>
</tr>
<tr>
<td>Signal state 1</td>
<td>Cancel alarm deleted active</td>
<td></td>
</tr>
<tr>
<td>Signal state 0</td>
<td>Cancel alarm deleted inactive</td>
<td></td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Applies only to HMI Advanced</td>
<td></td>
</tr>
</tbody>
</table>

Note: The signal is not reset automatically, it must be set by the user via the PLC user program.
### Z1: NC/PLC interface signals

#### 19.1 Various interface signals and functions (A2)

<table>
<thead>
<tr>
<th>DB19 DBX20.4</th>
<th>Recall alarm deleted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>Recall alarm deleted inactive</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>Recall alarm deleted inactive</td>
</tr>
<tr>
<td>Note: The signal is not reset automatically, it must be set by the user via the PLC user program.</td>
<td></td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Applies only to HMI Advanced</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB19 DBX20.6</th>
<th>Simulation active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>On entry to simulation = 1</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>On exit from simulation = 0</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Can be evaluated by machine manufacturer in order to activate the test on NC start.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB19 DBX20.7</th>
<th>Switch over MCS/WCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>The coordinate system is switched over from workpiece coordinate system (WCS) to machine coordinate system (MCS) or from MCS to WCS. After it has been set, the signal is active for one PLC cycle.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>No effect</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>The interface signal: DB19, DBX20.7 (switch over MCS/WCS) must be transferred to the interface signal: DB19, DBX0.7 (actual value in WCS) in order that the switchover becomes effective.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB19, DBX0.7 (actual value in WCS)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB19 DBB22</th>
<th>Displayed channel number from HMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB19 DBB24</th>
<th>Actual image number of the JobShop interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB19 DBX26.1</th>
<th>OK Part program handling status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>Transfer correctly completed</td>
</tr>
</tbody>
</table>
### DB19 DBX26.1 OK Part program handling status

<table>
<thead>
<tr>
<th>Signal state 0</th>
<th>Transfer completed with error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional references</td>
<td>Commissioning Manual, SINUMERIK Operate (IM9), Section &quot;PLC functions&quot; &gt; &quot;Program selection&quot;</td>
</tr>
</tbody>
</table>
| Corresponding to ... | DB19.DBB13 (program selection: Requirement)  
DB19.DBB16 (program selection: Control file index)  
DB19.DBB17 (program selection: Program list index)  
DB19.DBB27 (program selection: Error handling) |

### DB19 DBX26.2 Error (part program handling status)

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>Transfer completed with error</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>Transfer correctly completed</td>
</tr>
<tr>
<td>Additional references</td>
<td>Commissioning Manual, SINUMERIK Operate (IM9), Section &quot;PLC functions&quot; &gt; &quot;Program selection&quot;</td>
</tr>
</tbody>
</table>
| Corresponding to ... | DB19.DBB13 (program selection: Requirement)  
DB19.DBB16 (program selection: Control file index)  
DB19.DBB17 (program selection: Program list index)  
DB19.DBB27 (program selection: Error handling) |

### DB19 DBX26.3 Active (part program handling status)

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>Job in progress</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>No job in progress</td>
</tr>
<tr>
<td>Additional references</td>
<td>Commissioning Manual, SINUMERIK Operate (IM9), Section &quot;PLC functions&quot; &gt; &quot;Program selection&quot;</td>
</tr>
</tbody>
</table>
| Corresponding to ... | DB19.DBB13 (program selection: Requirement)  
DB19.DBB16 (program selection: Control file index)  
DB19.DBB17 (program selection: Program list index)  
DB19.DBB27 (program selection: Error handling) |

### DB19 DBX26.5 Unload (part program handling status)

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>Unload active</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>Unload inactive</td>
</tr>
<tr>
<td>Additional references</td>
<td>Commissioning Manual, SINUMERIK Operate (IM9), Section &quot;PLC functions&quot; &gt; &quot;Program selection&quot;</td>
</tr>
</tbody>
</table>
| Corresponding to ... | DB19.DBB13 (program selection: Requirement)  
DB19.DBB16 (program selection: Control file index)  
DB19.DBB17 (program selection: Program list index)  
DB19.DBB27 (program selection: Error handling) |
### DB19 DBX26.6

<table>
<thead>
<tr>
<th>Load (part program handling status)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong> No</td>
</tr>
<tr>
<td><strong>Signal state 1:</strong> Load active</td>
</tr>
<tr>
<td><strong>Signal state 0:</strong> Load inactive</td>
</tr>
<tr>
<td><strong>Additional references:</strong></td>
</tr>
<tr>
<td>Commissioning Manual, SINUMERIK Operate (IM9), Section &quot;PLC functions&quot; &gt; &quot;Program selection&quot;</td>
</tr>
<tr>
<td><strong>Corresponding to ...</strong></td>
</tr>
<tr>
<td>DB19.DBB13 (program selection: Requirement)</td>
</tr>
<tr>
<td>DB19.DBB16 (program selection: Control file index)</td>
</tr>
<tr>
<td>DB19.DBB17 (program selection: Program list index)</td>
</tr>
<tr>
<td>DB19.DBB27 (program selection: Error handling)</td>
</tr>
</tbody>
</table>

### DB19 DBX26.7

<table>
<thead>
<tr>
<th>Select (part program handling status)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong> No</td>
</tr>
<tr>
<td><strong>Signal state 1:</strong> Selection active</td>
</tr>
<tr>
<td><strong>Signal state 0:</strong> Selection inactive</td>
</tr>
<tr>
<td><strong>Additional references:</strong></td>
</tr>
<tr>
<td>Commissioning Manual, SINUMERIK Operate (IM9), Section &quot;PLC functions&quot; &gt; &quot;Program selection&quot;</td>
</tr>
<tr>
<td><strong>Corresponding to ...</strong></td>
</tr>
<tr>
<td>DB19.DBB13 (program selection: Requirement)</td>
</tr>
<tr>
<td>DB19.DBB16 (program selection: Control file index)</td>
</tr>
<tr>
<td>DB19.DBB17 (program selection: Program list index)</td>
</tr>
<tr>
<td>DB19.DBB27 (program selection: Error handling)</td>
</tr>
</tbody>
</table>

### DB19 DBB27

<table>
<thead>
<tr>
<th>Error program handling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong> No</td>
</tr>
<tr>
<td><strong>Signal(s) updated:</strong> Cyclically</td>
</tr>
<tr>
<td><strong>Description:</strong> Error codes: aaa</td>
</tr>
<tr>
<td><strong>Value</strong></td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td><strong>Additional references:</strong> Commissioning Manual, SINUMERIK Operate (IM9), Section &quot;PLC functions&quot; &gt; &quot;Program selection&quot;</td>
</tr>
<tr>
<td><strong>Corresponding to ...</strong></td>
</tr>
<tr>
<td>DB19.DBB13 (program selection: Requirement)</td>
</tr>
<tr>
<td>DB19.DBB16 (program selection: Control file index)</td>
</tr>
<tr>
<td>DB19.DBB17 (program selection: Program list index)</td>
</tr>
<tr>
<td>DB19.DBB26 (program selection: Acknowledgement)</td>
</tr>
</tbody>
</table>
### 19.1.6 Signals to channel (DB21, ...)

<table>
<thead>
<tr>
<th>DB21, ... DBX6.2</th>
<th>Delete distance-to-go (channel-specific)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>Delete distance-to-go (channel-specific): IS &quot;Delete distance-to-go (channel-specific)&quot; for path axes is only active in AUTOMATIC mode. The rising edge of the interface signal is only effective for the axes involved in the geometry grouping. These are also stopped with a ramp stop and their distance-to-go deleted (setpoint - actual value difference). Any remaining following error is still corrected. The next program block is then started. IS &quot;Delete distance-to-go (channel-specific)&quot; is therefore ignored by positioning axes. Remark: IS &quot;Delete distance-to-go&quot; does not influence the running dwell time in a program block with dwell time.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>No effect</td>
</tr>
<tr>
<td>Signal irrelevant for ...</td>
<td>Positioning axes</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>To terminate motion because of an external signal (e.g. measuring probe)</td>
</tr>
<tr>
<td>Special cases, errors, ....</td>
<td>Delete distance-to-go (channel-specific) When the axes have been stopped with IS &quot;Delete distance-to-go&quot; the next program block is prepared with the new positions. After a &quot;Delete distance-to-go&quot;, geometry axes thus follow a different contour to the one originally defined in the part program. If ( \text{G90} ) is programmed in the block after &quot;Delete distance-to-go&quot; it is at least possible to approach the programmed absolute position. On the other hand, with ( \text{G91} ), the position originally defined in the part program is no longer reached in the following block.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB31, ... DBX2.2 (delete distance-to-go (axis-specific))</td>
</tr>
</tbody>
</table>

### 19.1.7 Signals from channel (DB21, ...)

<table>
<thead>
<tr>
<th>DB21, ... DBX36.6</th>
<th>Channel-specific NCK alarm is active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>At least one NCK alarm is present for this channel. Thus the following group interface signal is also set: DB10 DBX109.0 (NCK alarm is present) The PLC user program can interrogate whether processing for the channel in question has been interrupted because of an NCK channel: DB21, ... DBX36.7 (NCK alarm with processing stop present).</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>No NCK alarm is active for this channel.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB21, ... DBX36.7 (NCK alarm with processing stop pending) DB10 DBX109.0 (NCK alarm pending)</td>
</tr>
<tr>
<td>Additional references</td>
<td>Diagnostics guide</td>
</tr>
</tbody>
</table>
### 19.1.8 Signals to axis/spindle (DB31, ...)

<table>
<thead>
<tr>
<th>DB31, ... DBX1.0</th>
<th>Drive test travel enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>
| Signal state 1 | Traversing motions of the axis are enabled for the drive test.  
Feedback signal of the safety handshake at the start of the NC function generator.  
The NC has requested the traversing enable for the axis at the start of the function generator with  
the following interface signal:  
DB31, ... DBX61.0 = 1 (drive test travel request)  
The PLC user program sets the current interface signal as feedback signal to the NC that the axis  
can be traversed:  
DB31, ... DBX1.0 = 1 (drive test travel enable)  
Only the PLC can decide on the traversing enable of an axis. |
<p>| Signal state 0 | Traversing motions of the axis are disabled for the drive test. |
| Additional references | CNC Commissioning Manual: NCK, PLC, Drive |</p>
<table>
<thead>
<tr>
<th></th>
<th>Axis/spindle disable</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB31, ...</td>
<td></td>
</tr>
<tr>
<td>DBX1.3</td>
<td></td>
</tr>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>
Axis/spindle disable

Signal state 1

**Axis**
- No setpoints are output to the position controller, i.e. the axis traversing movement is disabled. The axis is in closed-loop position control and any remaining following error is compensated.
- If the axis is traversed via an NC program or manually, the set position and set velocity are displayed as actual position and actual velocity on the user interface. With channel reset or end of program (M30/M2), the display of the actual value is set to the actual value of the machine axis.
- Travel commands are output to the NC/PLC interface.
- For a traversing axis, the axis is stopped in compliance with the braking characteristic and an alarm is displayed.

**Spindle**
- No setpoints are output on the speed controller during open-loop control mode.
- No setpoints are output on the position controller during positioning mode.
- The traversing motion of the spindle is disabled. The speed setpoint is displayed as the actual speed value on the user interface.
- If the interface signal is set for a rotating spindle, the spindle is stopped in compliance with the acceleration characteristic and an alarm is displayed.

**Structure graphic**

![Structure graphic](image)

**Note**
- If one of the following interface signals is set for a traversing axis/spindle for which "Axis/spindle disable" is active, the axis/spindle is not stopped:
  - DB21, ... DBX12.3, 16.3 and 20.3 (feedrate stop for geometry axes 1, 2 and 3)
  - DB21, ... DBX320.3, 324.3 and 328.3 (feedrate stop for orientation axes 1, 2 and 3)
  - DB31, ... DBX4.3 (feedrate stop / spindle stop)
- The axis/spindle can still be switched to the "hold" or "follow up" state with DB31, ... DBX1.4 (follow-up mode).
- If "Axis/spindle disable" is briefly set for a traversing axis, the axis is stopped without an alarm. With the next travel request, the axis is traversed to the programmed position.

**Example**

N10 G0 X0 Y0
N20 G1 F1000 X100
N30 Y100
N40 X200

Regarding N20: At position 20 mm,"Axis/spindle disable" is briefly set for axis X ⇒ axis X is stopped
Regarding N40: Axis X traverses from the last position (approx. 20 mm) to the programmed position 200 mm.

- Effects of "Axis/spindle disable" for spindle or axis couplings:

<table>
<thead>
<tr>
<th>LS/LA</th>
<th>FS/FA</th>
<th>Coupl.</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Off</td>
<td>Setpoints are output</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Off</td>
<td>No setpoint output for following spindle/axis</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Off</td>
<td>No setpoint output for leading spindle/axis</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Off</td>
<td>No setpoint output for leading and following spindle/axis</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>On</td>
<td>Setpoints are output</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>On</td>
<td>Axis/spindle disable has no effect on the FS/FA</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>On</td>
<td>Axis/spindle disable also has an effect on the FS/FA</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>On</td>
<td>No setpoint output</td>
</tr>
</tbody>
</table>

1) Status of the interface signal DB31, ... DBX99.0 (leading spindle/axis active)
2) Status of the interface signal DB31, ... DBX99.1 (following spindle/axis active)
3) Status of the coupling between leading and following spindle/axis

Signal state 0: The reset of the interface signal does not take effect until the axis/spindle is stationary.

- **Axis**
  - Setpoints are output on the position controller again immediately after the reset of the interface signal.

- **Spindle**
  - After the reset of the interface signal, the spindle disable which is still effective internally must be cancelled by an axis-specific reset, channel reset or end of program (M30/M2). Only then are setpoints output on the speed controller again.

Application example(s): Prevention of traversing motions of the axis/spindle when running-in an NC program.

Corresponding to ... DB21, ... DBX33.7 (program test active)

Additional references: Behavior in synchronous mode: Function Manual, Extended Functions; Synchronous Spindle (S3)

---

Follow-up mode

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
</table>

Signal state 1: Follow-up mode is active:

- The position setpoint is continuously tracked: Position setpoint = actual position value
- Feedback signal: DB31, ... DBX61.3 = 1 (follow-up active)
- Standstill and clamping monitoring are not active.
- If an NC program is active when the closed-loop control system is switched on again, a control-internal repositioning operation is performed (REPOSA: Linear approach with all axes) to the last programmed position.

**Note**

Follow-up mode is only useful with a simultaneous reset of the controller enable: DB31, ... DBX2.1 = 0
### Follow-up mode

**Signal state 0**

Follow-up mode is not active:
- The position setpoint is not tracked: Position setpoint = programmed set position
- Feedback signal: DB31, ... DBX61.3 = 0 (follow-up not active)
- Standstill and clamping monitoring are active.

**Note**

If the controller enable is reset (DB31, ... DBX2.1 == 0), the axis is in the "hold" state. Whereby the set position of the axis is not corrected to the actual position. If the axis is moved in this state, e.g. manually, the difference between the set position and the actual position increases constantly (following error). The following error is suddenly corrected to zero when the "Controller enable" is set (speed setpoint jump).

**Special cases, errors, ...**

If the controller enable is cancelled in the control because of faults, the "hold" state should be activated for the axis before the NC start after the queued alarms have been successfully deleted, and the accompanying controller enable set: DB31, ... DBX1.4 = 0 (follow-up mode).

Otherwise, for an NC Start and active follow-up mode, the traversing distance of the previous NC block would not be executed due to the internal delete distance-to-go.

**Notice:**

During the transition from the "follow-up" state to the "hold" state or when the controller enable is set in position control, delete distance-to-go is activated in the control. As a consequence, a traversing block in which only this axis is traversed, is terminated directly.

**Corresponding to ...**

- DB31, ... DBX2.1 (controller enable)
- DB31, ... DBX2.3 (clamping in progress)
- DB31, ... DBX61.3 (follow-up active)

---

### Position measuring system 1 (PMS1) / Position measuring system 2 (PMS2)

**Edge evaluation:** No

**Signal(s) updated:** Cyclically

- **Signal state 1** The position measuring system is active
- **Signal state 0** The position measuring system is inactive
### Position measuring system 1 (PMS1) / Position measuring system 2 (PMS2)

<table>
<thead>
<tr>
<th>Signal state overview</th>
<th>PMS1</th>
<th>PMS2</th>
<th>Effect</th>
</tr>
</thead>
</table>
| 1                     | 0    | 0    | Position measuring system 1 is active:  
  - Position control of the machine axis via position measuring system 1.  
  - Monitoring functions (measuring system, standstill, clamping monitoring, contour deviation, etc.) of the machine axis via position measuring system 1.  
  - If position measuring system 2 exists (MD30200 $MA_NUM_ENCS == 2), its actual position value is acquired, but not monitored by any of these functions. |
| 0                     | 1    | 1    | Position measuring system 2 is active:  
  - Position control of the machine axis via position measuring system 2.  
  - Monitoring functions (measuring system, standstill, clamping monitoring, contour deviation, etc.) of the machine axis via position measuring system 2.  
  - If position measuring system 1 exists (MD30200 $MA_NUM_ENCS == 2), its actual position value is acquired, but not monitored by any of these functions. |
| 1                     | 1    | 1    | Position control of the machine axis via position measuring system 1.  
  - If position measuring system 2 is available (MD30200 $MA_NUM_ENCS == 2), its actual position value is also acquired. |
| 0                     | 0    | 0    | Position measuring systems 1 and 2 are inactive ("parking" of the machine axis):  
  - There is no actual value acquisition.  
  - The monitoring of the position measuring system has been deactivated.  
  - The following interface signals are reset:  
    - DB31, ... DBX60.4/5 == 0 (referenced/synchronized encoder 1/2)  
    - DB31, ... DBX61.5 (position controller active)  
    - DB31, ... DBX61.6 (speed controller active)  
    - DB31, ... DBX61.7 (current controller active) |

**Notes**
- If the interface signal of the active position measuring system is reset for a traversing axis, the axis is stopped with a ramp stop without the controller enable being cancelled internally.
- If a speed-controlled spindle does not have a position measuring system, the "Controller enable" interface signal must be set:  
  DB31, ... DBX2.1 == 1 (controller enable)  
- After deactivation of the "parking" state, incremental position measuring systems have to be referenced to achieve the "referenced" encoder status.
### Application example(s)

1. **Switching over from position measuring system 1 to positioning measuring system 2 (and vice versa).**
   - If the axis was referenced in both position measuring systems and the limit frequency of the used measured value encoder has not been exceeded in the meantime, i.e. DB31, ... DBX60.4 and 60.5 == 1 (referenced/synchronized 1 and 2), a new reference point approach is not required after the switchover.
   - On switchover, the current difference between position measuring system 1 and 2 is traversed immediately.
   - A tolerance band in which the deviation between the two actual values may lie at the switchover can be specified in the following machine data:
     - MD36500 $MA\_ENC\_CHANGE\_TOL$ (maximum tolerance for actual position value switchover)
   - If the actual value difference is greater than the tolerance, switchover is not performed and alarm 25100 "Measuring system switchover not possible" is displayed.

2. **Machine axis is parked:**
   - The position measuring system monitoring is switched off when the measured value encoder is removed.

3. **Switch off position measuring system:**
   - When the position measuring system 1 or 2 is switched off, the associated interface signal is reset:
     - DB31, ... DBX60.4/5 (referenced/synchronized 1/2)

4. **Reference point approach:**
   - Reference point approach of the axis is executed with the selected position measuring system.
   - Every position measuring system must be referenced separately.

### Special cases, errors, ....

If the "parking" state is active, the following interface signal is ignored at NC start for this axis:
- DB31, ... DBX60.4 / 5 (referenced/synchronized 1 / 2).

### Corresponding to ...

- DB31, ... DBX60.4/.5 (referenced/synchronized 1/2)
- DB31, ... DBX61.6 (speed controller active)
- DB31, ... DBX2.1 (controller enable)
- MD36500 $MA\_ENC\_CHANGE\_TOL$ (max. tolerance on actual position value switchover)
- MD30200 $MA\_NUM\_ENCS$ (number of encoders)

### Additional references

- Function Manual, Basic Functions; Velocities, Setpoint / Actual Value Systems, Closed-Loop Control (G2)
### Signal state 1

**Controller is enabled.**  
The position control loop is closed and the axis/spindle is in closed-loop control.  
**Feedback:**  
\[ DB31, \ldots DBX61.5 = 1 \] (position controller active)  
If the axis/spindle was referenced before resetting the interface signal, the axis/spindle does not have to be re-referenced after the interface signal is set again. Supplementary condition: The limit frequency of the active measuring system must not be exceeded in the meantime.  
**Note**  
If the axis/spindle was moved from its position during the time in which the controller enable was not set, the behavior when the controller enable is set depends on the interface signal "follow-up mode":  
- \[ DB31, \ldots DBX1.4 == 1 \] (follow-up mode)  
  Position control is performed at the current position  
- \[ DB31, \ldots DBX1.4 == 0 \] (follow-up mode)  
  Position control is performed at the last position before the controller enable is reset

### Signal state 0

**Controller is not enabled.**  
The behavior when the "controller enable" is removed depends on whether the axis/spindle is stationary or traversing at this time:  
- **Axis/spindle stationary:**  
  - The position control loop of the axis is opened.  
  - For \[ DB31, \ldots DBX1.4 == 1 \] (follow-up mode) \( \Rightarrow \) position setpoint = actual position value  
  - The controller enable on the drive is reset  
  - The following interface signals are reset:  
    - \[ DB31, \ldots DBX61.5 = 0 \] (position controller active)  
    - \[ DB31, \ldots DBX61.6 = 0 \] (speed controller active)  
    - \[ DB31, \ldots DBX61.7 = 0 \] (current controller active)  
- **Axis/spindle traverses**  
  - The axis is stopped with rapid stop.  
  - Alarm 21612 "Controller enable VDI signal reset during motion".  
  - The position control loop of the axis/spindle is opened.  
  - Independent of the interface signal \[ DB31, \ldots DBX1.4 \] (follow-up mode), the position setpoint is corrected at the end of the braking operation (position setpoint = actual position value) and the feedback signal \[ DB31, \ldots DBX61.3 = 1 \] (follow-up mode) is set.  
  - The following interface signals are reset:  
    - \[ DB31, \ldots DBX61.5 \] (position controller active)  
    - \[ DB31, \ldots DBX61.6 \] (speed controller active)  
    - \[ DB31, \ldots DBX61.7 \] (current controller active)  

**Note**  
The current set position is retained in the control when "Controller enable" is reset and "Follow-up mode" \( DB31, \ldots DBX1.4 == 0 \) is not set. If the axis is moved in this state, e.g. manually, the difference between the set position and the actual position increases constantly (following error). The following error is suddenly corrected to zero when the "Controller enable" is set (speed setpoint jump).
Mechanical clamping of an axis

If the axis is positioned at the clamping position, the clamping is closed. The controller enable is then reset. Otherwise, the position controller would constantly work against the clamping, if the axis was moved mechanically from its specified position during the clamping operation.

When the clamping is removed, the controller enable is set first and then the mechanical clamping is released.

Travel request for an axis/spindle without controller enable:
- The axis/spindle is not traversed
- The travel command is output to the interface
- As long as the travel request is present, the axis/spindle is traversed immediately when the controller enable is set.

When the controller enable is reset for a traversing geometry axis, this always results in a contour violation.

Delete distance-to-go (axis-specific) / spindle reset

Delete distance-to-go (axis-specific) is requested. The behavior varies depending on the operating mode:

- JOG mode:
  If the interface signal is set for one axis (edge change 0 → 1), this axis is stopped with ramp stop and its distance-to-go deleted (setpoint - actual value difference). Any remaining following error is then corrected.

- AUTOMATIC and MDA modes:
  The rising edge of the interface signals only influences the axes which are not in the geometry grouping. They are stopped with ramp stop and their distance-to-go deleted (setpoint - actual value difference). The next program block can then be started. IS "delete distance-to-go axial" is therefore ignored by geometry axes.

Remark: "Delete distance-to-go" does not influence the running dwell time in a part program block with dwell time.

To terminate motion because of an external signal (e.g. measuring probe)
Delete distance-to-go (axis-specific) / spindle reset

Special cases, errors, ....

"Delete distance-to-go (axis-specific)"

After the axes have been stopped with "Delete distance-to-go" the next program block is prepared with the new positions. The axes thus follow a different contour to the one originally defined in the part program after a "Delete distance-to-go".

If G90 is programmed in the block after "Delete distance-to-go" it is at least possible to approach the programmed absolute position. On the other hand, with G91, the position originally defined in the part program is no longer reached in the following block.

Corresponding to ... DB21, ... DBX6.2 (delete distance-to-go, channel-specific)

Additional references Function Manual, Basic Functions; Spindles (S1)

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Signal irrelevant for ... MD35590 $MA_PARAMSET_CHANGE_ENABLE = 0

Corresponding to ... DB31, ... DBX9.0, .1, .2 (feedback: Active position controller parameter set)

Additional references CNC Commissioning Manual: NCK, PLC, Drive, Section "Commissioning NCK" >

Parameter set switchover commands from NC disabled

| Signal state 1 | The parameter set switchover is disabled. |
| Signal state 0 | The parameter set switchover is enabled. |

Corresponding to ... DB31, ... DBX9.0, .1, .2
### Ramp-function generator disable

| Signal state 1 | A rapid stop with speed setpoint 0 is requested for the drive. The drive is stopped without a ramp function (regenerative braking).
| Signal state 0 | No rapid stop with speed setpoint 0 is requested for the drive.

**Remark**
- A PROFIDrive telegram compatible with the "SIMODRIVE 611 universal" interface mode must be set in the drive.
- See drive parameters: p0922 and p2038

**Additional references**
- NC: CNC Commissioning Manual: NCK, PLC, Drive
- Drive: SINAMICS S120/S150 List Manual

### Motor/drive data set: Request interface

| Meaning | The switchover to a new motor (MDS) and/or to drive data sets (DDS) is requested via the interface: Formatting the request interface, i.e. which bits are used to address the motor data sets (MDS) – and which are used to address the drive data sets (DDS) is set via the formatting interface.
| Remark | For main spindle drives:
- MDS[ 0 ] → star operation
- MDS[ 1 ] → delta operation
| Application example(s) | For example, it is possible to switch between operating mode 1 (star operation) and operating mode 2 (delta operation) for a main spindle drive (MSD) by switching over the motor data set. It may be necessary to switch over the drive parameter set, for:
- Gear stage change
- Changing the measuring circuit
| Special cases, errors, .... | In principle it is possible to switch over drive parameter sets at any time. However, as torque jumps can occur when switching over speed controller parameters and motor speed normalization, parameters should only be switched over in stationary states, especially axis standstill. As soon as the request to switch over to another motor data set is identified, the pulses enable is reset.

**Corresponding to**
- DB31, ... DBX92.1 (ramp-function generator disable active)
- DB31, ... DBX93.0 - 4 (motor/drive data set: display interface)
- DB31, DBX130.0 - 4 (motor/drive data set: formatting interface)
- DB31, ... DBX21.5 (motor being selected)

**Additional references**
- Function Manual, Basic Functions, Chapter "A2: Various NC/PLC interface signals and functions" 
  > "Switchover motor/drive data sets"
- CNC Commissioning Manual: NCK, PLC, Drive
### DB31, ... DBX21.5
#### Motor being selected

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal state 1</strong></td>
<td>The necessary electrical and/or mechanical switchover operations (e.g. contactor switchover for star-delta switchover) have been completed.&lt;br&gt;The axis can be traversed again. The pulses are then enabled by the drive.&lt;br&gt;&lt;strong&gt;Note&lt;/strong&gt;: The interface signal must be reset before a request to switch over to a new motor data set (MDS) and/or drive data set (DDS) (DB31, ... DBX21.0 - 4).</td>
</tr>
<tr>
<td><strong>Signal state 0</strong></td>
<td>The necessary electrical (e.g. contactor switchover for star-delta switchover) and/or mechanical switchovers are <strong>not</strong> complete. The axis must not traverse. The pulses are not enabled by the drive.</td>
</tr>
</tbody>
</table>

**Corresponding to ...**
- DB31, ... DBX21.0 - 4 (motor/drive data set: request interface)
- DB31, ... DBX93.0 - 4 (motor/drive data set: display interface)
- DB31, ... DBX130.0 - 4 (motor/drive data set: formatting interface)

**Additional references**
- Function Manual, Basic Functions, Chapter "A2: Various NC/PLC interface signals and functions" > "Switchover motor/drive data sets"
- CNC Commissioning Manual: NCK, PLC, Drive

### DB31, ... DBX21.6
#### Integrator disable, speed controller

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal state 1</strong></td>
<td>The integrator (I component) of the speed controller is disabled or is to be disabled (P instead of PI behavior).&lt;br&gt;&lt;strong&gt;Note&lt;/strong&gt;: If the speed controller integrator disable is activated, compensation operations may occur in certain applications (e.g. if the integrator was already holding a load while stationary).&lt;br&gt;The drive acknowledges the integrator disable to the PLC using the interface signal: DB31, ... DBX93.6 (speed controller integrator disabled).</td>
</tr>
<tr>
<td><strong>Signal state 0</strong></td>
<td>The integrator (I component) of the speed controller is enabled (PI behavior).</td>
</tr>
</tbody>
</table>

**Corresponding to ...**
- DB31, ... DBX93.6 (feedback: The speed controller integrator is disabled)

**Additional references**
- CNC Commissioning Manual: NCK, PLC, Drive

### DB31, ... DBX21.7
#### Pulse enable

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal state 1</strong></td>
<td>The pulses are enabled for the drive.&lt;br&gt;The pulse enable is only performed in the drive when the drive signals readiness: DB31, ... DBX93.5 == 1 (feedback: Drive ready)</td>
</tr>
<tr>
<td><strong>Signal state 0</strong></td>
<td>The pulses are disabled for the drive.</td>
</tr>
</tbody>
</table>

**Application example(s)**
- Signal relevant to safety

**Special cases, errors, ....**
- If the pulse enable is removed during motion (e.g. emergency stop), the axis/spindle is no longer braked under control. The axis coasts to standstill.
19.1 Various interface signals and functions (A2)

19.1.9 Signals from axis/spindle (DB31, ...)

**Note**

**SINAMICS S120: Message word (MELDW)**

The message word (MELDW) is only contained in PROFIdrive telegrams compatible with SIMODRIVE 611U, for example, telegrams 102, 103, 105, 106, 110, 111, 116, 118, 125, 126, 136, 138, 139

**References:**

SINAMICS List Manual, Function Diagrams 2419 and 2420

**SINAMICS S120: Status word 1/2 (ZSW1/2)**

The status words ZSW1 or ZSW2 only refer to SIMODRIVE 611U compatible PROFIdrive telegrams in the following (SIMODRIVE 611U interface mode, p2038 = 1)

<table>
<thead>
<tr>
<th>DB31, ... DBX61.0</th>
<th>Drive test travel request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>

**Signal state 1**

The control signals that all of the traversing conditions for the drives are fulfilled.

Requirements for this are:

- The mechanical brake of the axis involved was previously released and all other axis traversing conditions are fulfilled.

  With:
  - DB31, ... DBX61.0 (drive test, travel request) = 1 signal
  - the appropriate axes can be moved.

- The axis disable:
  - DB31, ... DBX1.3 (axis/spindle disable) = 1 signal
  - is not active.

**Signal state 0**

The control signals that the axes cannot be moved.

Axes cannot be moved when:

- DB31, ... DBX61.0 (drive test, travel request) = 0 signal
- Faults are present in the control

This means that the requirements specified above are not fulfilled.

**Additional references**

Commissioning Manual IBN CNC: NCK, PLC, Drive
Follow-up active (feedback)

Signal state 1

The control signals that the follow-up mode for the axis/spindle is not active.

Requirements for this are:

- The controller enable for the drive has been withdrawn (either by the PLC with "controller enable \( = 0 \) signal or inside the control for faults; refer to the references)
- Follow-up operation is selected (either by the PLC with IS "follow-up operation" = 1 signal or in the control, e.g. when withdrawing the controller enable from an axis that is moving)

The position setpoint continually tracks the actual value while the follow-up mode is active.

Zero speed and clamping monitoring are not active.

Signal state 0

The control signals that follow-up mode for the axis/spindle is not active.

Zero speed and clamping monitoring are active.

This means that the requirements specified above are not fulfilled.

In the "hold" state, the interface signal: DB31, ... DBX61.3 (follow-up active) is 0.

Special cases, errors, ....

Notice:

A delete distance-to-go is triggered internally in the control on transition from "Follow up" to "Hold" (IS "Follow-up mode" = 0) or in the closed-loop control mode (IS "Controller enable" = 1).

Corresponding to ...

DB31, ... DBX2.1 (controller enable)
DB31, ... DBX1.4 (follow-up mode)

Additional references
Diagnostics Manual

Axis/spindle stationary \((n < \text{n}_{\text{min}})\) (status)

Signal state 1

The current speed of the axis or the actual number of rotations of the spindle lies below the limit given by the machine data:

\[
\text{MD36060} \ $\text{MA}_\text{STANDSTILL}_\text{VELO}_\text{TOL} \\
(\text{maximum velocity/speed for signal "Axis/spindle stationary")}.
\]

Signal state 0

The current velocity of the axis or the actual spindle speed is greater than the value specified in the MD (standstill range).

If a travel command is present, e.g. for a spindle, then the signal is always \( = 0 \) - even if the actual speed lies below that specified in MD36060.

If the interface signal:

DB31, ... DBX61.4 (axis/spindle stationary)

is signaled and there is no closed-loop position control active for the spindle, then at the MMC, an actual speed of zero is displayed and with the system variable \$\text{AA}_S[n]\) zero is read.

Application example(s)

Enable signal for opening a protective device (e.g. open door).

The workpiece chuck or the tool clamping device is only opened when the spindle is stationary.

The oscillation mode can be switched-on during gear stage change after the spindle has been braked down to standstill.

The tool clamping device must have been closed before the spindle can be accelerated.

Corresponding to ...

MD36060 $\text{MA}_\text{STANDSTILL}_\text{VELO}_\text{TOL} \\
(\text{maximum velocity/speed for signal "Axis/spindle stationary")}
### Z1: NC/PLC interface signals

#### 19.1 Various interface signals and functions (A2)

<table>
<thead>
<tr>
<th>DB31, ... DBX61.5</th>
<th>Position controller active (status)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td><strong>Signal state 1</strong></td>
<td>The control signals that the position controller for the axis or spindle is closed.</td>
</tr>
<tr>
<td><strong>Signal state 0</strong></td>
<td>The control signals that the position controller for the axis or spindle is open. If &quot;controller enable&quot; is canceled because of a fault or from the PLC user program the position controller is opened and therefore the interface signal DB31, ... DBX61.5 (position controller active) is set to 0. Spindle without position control: Signal &quot;Position controller active&quot; is always &quot;0&quot;. See References for other effects.</td>
</tr>
</tbody>
</table>

**Application example(s)**

If the position control is active the axis/spindle is kept in position by the position controller. Any brakes or clamps can thus be opened.

The interface signal:
DB31, ... DBX61.5 (position controller active)
can be used as feedback signal for the interface signal:
DB31, ... DBX2.1 (controller enable).
The holding brake of a vertical axis must be activated as soon as the position control is no longer active.

If a spindle has been technically designed/dimensioned for the purpose, in the part program, it can be changed-over into the closed-loop position controlled mode as spindle or as axis (with SPCON or $70).

In these cases, the interface signal "position controller active" is set.

**Special cases, errors, ....**

Special case for simulation axes (MD30350 $MA_SIMU_AX_VDI_OUTPUT = "1"):
The IS "position controller active" is also set for simulation axes as soon as MD = "1".

**Corresponding to ...**

DB31, ... DBX2.1 (controller enable)
DB31, ... DBX1.4 (follow-up mode)
DB31, ... DBX1.5 and 1.6 (position measuring system 1 and 2)

**Additional references**

Diagnostics Manual

<table>
<thead>
<tr>
<th>DB31, ... DBX61.6</th>
<th>Speed controller active (status)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td><strong>Signal state 1</strong></td>
<td>The control signals that the speed controller is closed for the axis or spindle.</td>
</tr>
<tr>
<td><strong>Signal state 0</strong></td>
<td>The control signals that the speed controller is open for the axis or spindle. The speed controller output is cleared.</td>
</tr>
</tbody>
</table>

**Application example(s)**

If the spindle is not under position control, the interface signal can be used as a feedback for the interface signal DB31, ... DBX2.1 (controller enable).

**Special cases, errors, ....**

The interface signal is also set for simulation axes:
Simulation axis: MD30350 $MA_SIMU_AX_VDI_OUTPUT == 1

**Corresponding to ...**

DB31, ... DBX61.5 (position controller active)
### Current controller active (status)

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal state 1</strong></td>
<td>The current controller for the axis/spindle is active.</td>
</tr>
<tr>
<td><strong>Signal state 0</strong></td>
<td>The current controller for the axis/spindle is not active. The current controller output, including the injection variables on the control voltage, is cleared.</td>
</tr>
</tbody>
</table>

**Corresponding to ...**

- DB31, ... DBX61.5 (position controller active)
- DB31, ... DBX61.6 (speed controller active)

### Active position controller parameter set A, B, C (feedback)

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: After switchover</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Meaning</strong></td>
<td>The active position controller parameter sets are displayed with the interface signals A, B and C. The following assignment applies:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Signal irrelevant for ...**

- MD35590 $MA_PARAMSET_CHANGE_ENABLE == 0

If the switchover of the position controller parameter set is switched off, the first parameter set is always active.

**Corresponding to ...**

- DB31, ...DBX9.0 - DBX9.2 (position controller parameter set selection A, B, C)

### Lubrication pulse

<table>
<thead>
<tr>
<th>Edge evaluation: Yes</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge change</strong></td>
<td>As soon as the axis/spindle has moved through the traversing distance set in machine data: MD33050 $MA_LUBRICATION_DIST (traversing distance for lubrication from the PLC), the interface signal is inverted.</td>
</tr>
<tr>
<td>0 → 1 or 1 → 0</td>
<td>Note: The distance measurement is restarted each time that the control runs up.</td>
</tr>
</tbody>
</table>

**Application example(s)**

- The lubrication pump for the axis/spindle can be activated with the "Lubrication pulse" interface signal. Machine bed lubrication therefore depends on the distance traveled.

**Corresponding to ...**

- MD33050 $MA_LUBRICATION_DIST (lubrication pulse distance)
Z1: NC/PLC interface signals

19.1 Various interface signals and functions (A2)

<table>
<thead>
<tr>
<th>DB31, ... DBX92.1</th>
<th>Ramp-function generator disable active (feedback)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>Ramp function generator fast stop is active for the drive.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>Ramp function generator fast stop is not active for the drive.</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Position controller bypassing of the ramp function generator.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB31, ... DBX20.1 (ramp-function generator fast stop)</td>
</tr>
<tr>
<td>Remark</td>
<td>In the SINAMICS a protocol can be set that runs in what is known as the 611U compatibility mode.</td>
</tr>
<tr>
<td>Additional references</td>
<td>Commissioning Manual IBN CNC: NCK, PLC, Drive</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB31, ... DBX92.4</th>
<th>Drive-autonomous motion active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>A drive-autonomous motion is active. The axis traverses due to setpoints created by drive-internal functions. The drive still responds to control signals of the NC, e.g. controller enable. Setpoint specifications of the NC are ignored.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>A drive-autonomous motion is not active.</td>
</tr>
</tbody>
</table>
| Application example(s) | Internal drive functions:  
  ● Rotor or pole position identification  
  ● Function generator |
| Remark              | DBX92.4 = 1 IF MELDW.11 == 1 (controller enable) AND ZSW1.2 == 0 (operation enabled) |
| Additional references | SINAMICS S120 Function Manual;  
  Pole position identification: Section "Servo control" > "Pole position identification"  
  Function generator: Section "Servo control" > "Optimization of the current and speed controller" |

<table>
<thead>
<tr>
<th>DB31, ... DBX93.0 - DBX93.4</th>
<th>Motor/drive data set: Display interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Meaning</td>
<td>The active motor data set (MDS) and drive data set (DDS) are displayed via the interface. Formatting the request interface, i.e. which bits are used to address the motor data sets (MDS) – and which are used to address the drive data sets (DDS) is set via the formatting interface.</td>
</tr>
</tbody>
</table>
| Remark                       | For main spindle drives:  
  ● MDS[ 0 ] → star operation  
  ● MDS[ 1 ] → delta operation |
| Corresponding to ...         | DB31, ... DBX21.0 - 4 (motor/drive data set: request interface)  
  DB31, ...DBX130.0 - 4 (motor/drive data set: formatting interface)  
  DB31, ... DBX21.5 (motor being selected) |
| Additional references        | Commissioning Manual IBN CNC: NCK, PLC, drive |
### DB31, DBX93.5
Drive ready

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>The drive is ready.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>The drive is <strong>not</strong> ready.</td>
</tr>
</tbody>
</table>

If the signal is reset while operational, the drive is stopped (pulse inhibit or fast stop). When powering-up, the pulses are still inhibited (canceled). In addition, the following interface signals are reset:
- DB10, DBX108.6 = 0 (drive ready)
- DB31, ..., DBX61.7 = 0 (current controller active)
- DB31, ..., DBX61.6 = 0 (speed controller active)

**Remark**
Interface signal: DB31, ..., DBX93.5 = drive: MELDW.12

**Corresponding to**
DB10, DBX108.6 (drives ready)
DB31, ..., DBX61.7 (current controller active)
DB31, ..., DBX61.6 (speed controller active)

**Additional references**
- Function Manual, Basic Functions, Chapter "A2: Various NC/PLC interface signals and functions"
- "Switchover motor/drive data sets"
- Commissioning Manual IBN CNC: NCK, PLC, drive
- SINAMICS S120/S150 List Manual

### DB31, DBX93.6
Speed controller integrator disabled (feedback)

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>The requested shutdown of the speed controller integrator is active in the drive. The speed controller has been switched over from the PI to P control response.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>The integrator of the speed controller is enabled. The speed controller functions as a PI controller.</td>
</tr>
</tbody>
</table>

**Corresponding to**
DB31, ..., DBX21.6 (integrator disable, n-controller)

**Additional references**
- Commissioning Manual IBN CNC: NCK, PLC, Drive

### DB31, DBX93.7
Pulses enabled (feedback)

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>The pulses of the assigned drive are enabled for the axis/spindle.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>The pulses of the assigned drive are not enabled.</td>
</tr>
</tbody>
</table>

If the pulses on the assigned drive are deleted, the following interface signals are reset:
- DB31, ..., DBX61.7 (current controller active)
- DB31, ..., DBX61.6 (speed controller active)
- DB31, ..., DBX61.5 (position controller active)

**Remark**
DB31, ..., DBX93.7 = MELDW.13

**Corresponding to**
DB31, ..., DBX21.7 (pulse enable)

**Additional references**
- Commissioning Manual IBN CNC: NCK, PLC, Drive
### Motor temperature prewarning

<table>
<thead>
<tr>
<th>DB31, ...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DBX94.0</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
</table>

**Signal state 1**
The motor temperature has exceeded the warning threshold (p0604) configured in the drive.

*Note*
If the motor temperature remains too high for longer than the parameterized time (p0606), a fault is output, the drive is stopped and the pulse enable removed.

If the motor temperature falls below the warning threshold (p0604) again before the time has expired (p0606), the interface signal is reset.

**Signal state 0**
The motor temperature is below the warning threshold (p0604).

**Remark**
The current motor temperature is displayed on the user interface at:
Operating area “Diagnostics” > “Service display: Axis/spindle”

**Corresponding to ...**
DB31, ... DBX94.1 (heat sink temperature prewarning)

**Additional references**
See DB31, ... DBX94.1 (heat sink temperature prewarning)

### Heat sink temperature prewarning

<table>
<thead>
<tr>
<th>DB31, ...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DBX94.1</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
</table>

**Signal state 1**
The heat sink temperature of the power semiconductors has exceeded the parameterized warning threshold (p0294).

*Note*
The parameterized reaction (p0290) is performed in the drive. If the temperature violation remains, a fault is output after approx. 20 s, the drive is stopped and the pulse enable removed.

**Signal state 0**
The drive module heat sink temperature monitoring has not responded, i.e. the temperature is below the warning threshold.

**Remark**
The interface signals DB31, ... DBX94.0 and .1 are derived from the following signals of the cyclic drive telegram:
- **Case 1:** Temperature warning in the message word
  - DB31, ... DBX94.0 ≠ MELDW, bit 6 (no motor overtemperature warning)
  - DB31, ... DBX94.1 ≠ MELDW, bit 7 (no thermal overload in power unit warning)
- **Case 2:** Warning of warning class B (interface mode "SIMODRIVE 611U", p2038 = 1)
  - DB31, ... DBX94.0 == 1 and DBX94.1 == 1, if the following applies:
    - Cyclic drive telegram, ZSW1: Bit 11 == 0 and 12 == 1 (warning class B)
  - The interface signals are derived from the warning of warning class B if there is no specific information from the message word.
  - An alarm is displayed. Alarm number = 200.000 + alarm value (r2124)

**Additional references**
- S120 Commissioning Manual, Section "Commissioning" > "Temperature sensors for SINAMICS components"
- S120 Function Manual, Section "Monitoring and protective functions"
- S120 List Manual
  - MELDW, bit 6 ≠ BO: r2135.14 → function diagram: 2548, 8016
  - MELDW, bit 7 ≠ BO: r2135.15 → function diagram: 2548, 2452, 2456, 8016
- SINUMERIK Diagnostics Manual
### Z1: NC/PLC interface signals

#### 19.1 Various interface signals and functions (A2)

| **DB31, ...**
| **DBX94.2** | **Run-up completed** |
| **Signal state 1** | The actual speed value has reached the speed tolerance band specified via p2164 after a new speed setpoint specification and has not left the band for the duration of p2166. Any subsequent speed fluctuations, also outside the tolerance band, e.g. due to load changes, will not affect the interface signal. |
| **Signal state 0** | The run-up procedure is still active after the speed setpoint has been changed. |
| **Corresponding to** | DB31, ... DBX94.6 ("n\_act = n\_set")
| | DB31, ... DBX94.3 ("|M_d| = M_{dx}\") |
| **Additional references** | • SINUMERIK Commissioning Manual IBN CNC: NCK, PLC, Drive
| | • SIMATIC S120 List Manual |

| **DB31, ...**
| **DBX94.3** | **(|M_d| < M_{dx})** |
| **Signal state 1** | The current torque utilization is below the torque utilization threshold (torque threshold 2, p2194). The run-up procedure is completed, the drive is in the steady state and the torque setpoint |M_d| drive does not exceed the threshold torque M_{dx}. The torque threshold characteristic is speed-dependent. During the run-up, DB31, ... DBX94.3 (|M_d| < M_{dx}) == 1. The interface signal is not updated until the run-up has been completed (DB31, ... DBX94.2 == 1) and the signal interlocking time for the threshold torque has expired. |
| **Signal state 0** | The torque setpoint |M_d| is larger than the threshold torque M_{dx}. A motor overload can be determined via the interface signal. An appropriate response can then be initiated in the PLC user program. |
| **Remark** | DB31, ... DBX94.3 = MELDW.1 |
| **Additional references** | • SINUMERIK Commissioning Manual IBN CNC: NCK, PLC, Drive
| | • SIMATIC S120 List Manual |

| **DB31, ...**
| **DBX94.4** | **(|n\_act| < n_{min})** |
| **Signal state 1** | The actual speed value n\_act is less than n_{min} (speed threshold value 3, p2161). |
| **Signal state 0** | The actual speed value is greater than the threshold minimum speed n_{min}. |
| **Remark** | DB31, ... DBX94.4 = MELDW.2 |
| **Additional references** | Commissioning Manual IBN CNC: NCK, PLC, Drive |

| **DB31, ...**
| **DBX94.5** | **(|n\_act| < n_{x})** |
| **Signal state 1** | The actual speed value n\_act is less than n_{x} (speed threshold value 2, p2155). |
| **Signal state 0** | The actual speed value n\_act is greater than the threshold speed n_{x}. |
### DB31, ..., DBX94.5

<table>
<thead>
<tr>
<th>DBX94.5</th>
<th>[n_{act} &lt; n_x]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remark</td>
<td>DB31, ..., DBX94.5 = MELDW.3</td>
</tr>
<tr>
<td>Additional references</td>
<td>Commissioning Manual IBN CNC: NCK, PLC, Drive</td>
</tr>
</tbody>
</table>

### DB31, ..., DBX94.6

<table>
<thead>
<tr>
<th>DBX94.6</th>
<th>[n_{act} = n_{set}]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>The actual speed value is at least the parameterized time (switch-on delay (n_{act} = n_{set}), p2167) within the tolerance band around the speed setpoint (speed threshold value 4, p2163).</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>The actual speed value is outside the tolerance band around the speed setpoint (speed threshold value 4, p2163).</td>
</tr>
<tr>
<td>Additional references</td>
<td>Commissioning Manual IBN CNC: NCK, PLC, Drive</td>
</tr>
</tbody>
</table>

### DB31, ..., DBX94.7

<table>
<thead>
<tr>
<th>Variable signaling function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
</tr>
<tr>
<td>Signal state 1</td>
</tr>
<tr>
<td>Signal state 0</td>
</tr>
<tr>
<td>Remark</td>
</tr>
<tr>
<td>Additional references</td>
</tr>
</tbody>
</table>

### DB31, ..., DBX95.7

<table>
<thead>
<tr>
<th>Warning of warning class C is pending</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
</tr>
<tr>
<td>Signal state 1</td>
</tr>
<tr>
<td>Signal state 0</td>
</tr>
<tr>
<td>Remark</td>
</tr>
<tr>
<td>Additional references</td>
</tr>
</tbody>
</table>

### DB31, ..., DBX130.0 - DBX21.4

<table>
<thead>
<tr>
<th>Motor/drive data set: Formatting interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
</tr>
<tr>
<td>Meaning</td>
</tr>
</tbody>
</table>
| Corresponding to ... | DB31, ..., DBX21.0 - 4 (motor/drive data set: request interface)  
DB31, ..., DBX93.0 - 4 (motor/drive data set: display interface) |
| Additional references |  
- Function Manual, Basic Functions, Chapter "A2: Various NC/PLC interface signals and functions" > "Switchover motor/drive data sets"  
- Commissioning Manual IBN CNC: NCK, PLC, drive |
19.2  Axis monitoring, protection zones (A3)

19.2.1  Signals to channel (DB21, ...)

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>Enable protection zones</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX1.1</td>
<td></td>
</tr>
</tbody>
</table>

**Edge evaluation:** Yes  
**Signal(s) updated:** Cyclically

**Signal state 1 or edge change 0 → 1**  
When a positive edge of this signal appears, a protection zone is enabled and the active alarm cleared. Then, motion can start in the same protection zone. As a result of the start of motion, the protection zone is enabled, the IS "machine or channel-specific protection zone violated" is set, and the axis starts to move. The enabling signal is canceled if motion is started that does not lead into the enabled protection zone.

**Signal state 0 or edge change 1 → 0**  
No effect

**Application example(s)**  
This allows protection zones to be released:
- If the current position is within a protection zone (alarm 2 present)
- If motion is to be started towards the protection zone limit (alarm 1 or 2 present)

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>Activate machine-specific protection zone 1 ( ...10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX8.0 - DBX9.1</td>
<td></td>
</tr>
</tbody>
</table>

**Edge evaluation:** No  
**Signal(s) updated:** Cyclically

**Signal state 1 or edge change 0 → 1**  
The pre-activated, machine-related protection zone 1 ( ...10) is activated by the PLC user program. The protection zone is immediately activated. Only protection zones that have been pre-activated in the part program can be activated.

**Signal state 0 or edge change 1 → 0**  
The pre-activated, machine-related protection zone 1 ( ...10) is de-activated by the PLC user program. The protection zone is immediately de-activated. Only protection zones that have been activated via the PLC and have been pre-activated in the NC part program can be de-activated.

**Application example(s)**  
Before a sensor, for example, is moved into the working range, the relevant machine-related protection zone can be activated.

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>Activate channel-specific protection zone 1 ( ...10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX10.0 - DBX11.1</td>
<td></td>
</tr>
</tbody>
</table>

**Edge evaluation:** No  
**Signal(s) updated:** Cyclically

**Signal state 1 or edge change 0 → 1**  
The pre-activated, channel-specific protection zone 1 ( ...10) is activated by the PLC user program. The protection zone is immediately activated. Only protection zones that have been pre-activated in the part program can be activated.
### 19.2.2 Signals from channel (DB21, ...

<table>
<thead>
<tr>
<th>Z1: NC/PLC interface signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.2 Axis monitoring, protection zones (A3)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB21, ...) DBX10.0 - DBX11.1</th>
<th><strong>Activate channel-specific protection zone 1 (...10)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal state 0 or edge change 0 → 0</strong></td>
<td>The pre-activated, channel-specific protection zone 1 (...10) is de-activated by the PLC user program. The protection zone is immediately de-activated. Only protection zones that have been activated via the PLC and have been pre-activated in the NC part program can be de-activated.</td>
</tr>
<tr>
<td><strong>Application example(s)</strong></td>
<td>Before a synchronous spindle, for example, is moved into the working range, the relevant machine-related protection zone can be activated.</td>
</tr>
</tbody>
</table>

**19.2.2 Signals from channel (DB21, ...)**

<table>
<thead>
<tr>
<th>DB21, ...) DBX272.0 – DBX273.1</th>
<th><strong>Machine-related protection zone 1 (...10) pre-activated</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation: No</strong></td>
<td><strong>Signal(s) updated: Cyclically</strong></td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>The machine-related protection zone 1 (...10) is pre-activated in the current block. (Pre-activated in the part program). The protection zone can therefore be activated or de-activated in the PLC user program using the interface signal: DB21, ...) DBX8.0 - DBX9.1 (machine-related protection zone 1 (...10))</td>
</tr>
<tr>
<td><strong>Signal state 0 or edge change 1 → 0</strong></td>
<td>The machine-related protection zone 1 (...10) is deactivated in the current block. (De-activated in the part program). The protection zone can therefore not be activated or de-activated in the PLC user program using the interface signal: DB21, ...) DBX8.0 to DBX9.1 (activate machine-related protection zone 1 (...10))</td>
</tr>
</tbody>
</table>

**Corresponding to ....** | DB21, ...) DBX8.0 - DBX9.1 (activate machine-related protection zone 1 (...10)) |

<table>
<thead>
<tr>
<th>DB21, ...) DBX274.0 – DBX275.1</th>
<th><strong>Channel-specific protection zone 1 (...10) pre-activated</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation: No</strong></td>
<td><strong>Signal(s) updated: Cyclically</strong></td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>The channel-specific protection zone 1 (...10) is pre-activated in the current block. (Pre-activated in the part program). The protection zone can therefore be activated or de-activated in the PLC user program using the interface signal: DB21, ...) DBX10.0 - DBX11.1 channel-specific protection zone 1 (...10))</td>
</tr>
<tr>
<td><strong>Signal state 0 or edge change 1 → 0</strong></td>
<td>The channel-specific protection zone 1 (...10) is deactivated in the current block. (De-activated in the part program.) The protection zone can therefore not be activated or de-activated in the PLC user program using the interface signal: DB21, ...) DBX10.0 - DBX11.1 (channel-specific protection zone 1 (...10))</td>
</tr>
</tbody>
</table>

**Corresponding to ....** | DB21, ...) DBX10.0 - DBX11.1 (activate channel-specific protection zone 1 (...10)) |
### Machine-related protection zone 1 (...10) violated

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>The activated, machine-related protection zone 1 (...10) is violated in the current block or in the current JOG movement. The pre-activated, machine-related protection zone 1 (...10) would be violated in the current block if it would be activated by the PLC.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>The activated, machine-related protection zone 1 (...10) is not violated in the current block. The pre-activated, machine-related protection zone 1 (...10) would not be violated in the current block if it would be activated by the PLC.</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Before parts are moved into the working zone - this IS can be used to check as to whether the tool or workpiece is located in the machine-related protection zone of the part to be moved in.</td>
</tr>
</tbody>
</table>

### Channel-specific protection zone 1 (...10) violated

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>The activated, channel-specific protection zone 1 (...10) is violated in the current block. The pre-activated, channel-specific protection zone 1 (...10) would be violated in the current block if it would be activated by the PLC.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>The activated, channel-specific protection zone 1 (...10) is not violated in the current block. The pre-activated, channel-specific protection zone 1 (...10) would not be violated in the current block if it would be activated by the PLC.</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Before parts are moved into the working zone - this IS can be used to check whether the tool or workpiece is located in the channel-specific protection zone of the part to be moved in.</td>
</tr>
</tbody>
</table>

### 19.2.3 Signals to axis/spindle (DB31, ...)

<table>
<thead>
<tr>
<th>DB31, ... DBX2.3</th>
<th>Clamping in progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>Clamping in progress. The clamping monitoring function is activated.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Clamping completed. The clamping monitoring function is replaced by the standstill (zero speed) monitoring.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>MD36050 $MA_CLAMP_POS_TOL (clamping tolerance)</td>
</tr>
</tbody>
</table>
**Velocity/spindle speed limitation**

<table>
<thead>
<tr>
<th>Edge evaluation: no</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>The NCK limits the velocity/spindle speed to the limit value set in the machine data: MD35160 $MA_SPIND_EXTERN_VELO_LIMIT</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>No limitation active.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>MD35100 $MA_SPIND_VELO_LIMIT (max. spindle speed) SD43220 $SA_SPIND_MAX_VELO_G26 (programmable upper spindle speed limitation for G26) SD43230 $SA_SPIND_MAX_VELO_LIM (spindle speed limitation for G96/G961/G97)</td>
</tr>
</tbody>
</table>

**Hardware limit switches plus and minus**

<table>
<thead>
<tr>
<th>Edge evaluation: no</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>A switch can be mounted at each end of the travel range of a machine axis which will cause a signal &quot;hardware limit switch plus or minus&quot; to be signaled to the NC via the PLC if it is actuated. If the signal is recognized as set, alarm 021614 &quot;hardware limit switch + or -&quot; is output and the axis is decelerated immediately. The braking/deceleration type is defined using the machine data: MD36600 $MA_BRAKE_MODE_CHOICE (braking behavior at the hardware limit switch) If the controller enable is withdrawn at the same time as the &quot;hardware limit switch&quot; signal, then the axis responds as described in Chapter A2 (&quot;various interface signals&quot;).</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Normal condition - a hardware limit switch has not been actuated.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>MD36600 $MA_BRAKE_MODE_CHOICE (deceleration behavior when the hardware limit switch responds)</td>
</tr>
</tbody>
</table>

**2nd software limit switch plus or minus**

<table>
<thead>
<tr>
<th>Edge evaluation: no</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>2nd software limit switch for the plus and minus directions is effective. 1st software limit switch for the plus and minus directions is not effective. In addition to the 1st software limit switches (plus and minus), the 2nd software limit switches (plus and minus) can be activated using these interface signals. The position is defined using machine data: MD36130 $MA_POS_LIMIT_PLUS2 (2nd software limit switch plus) and MD36120 $MA_POS_LIMIT_MINUS2 (2nd software limit switch minus)</td>
</tr>
</tbody>
</table>

19.2.4 Signals from axis/spindle (DB31, ...)

<table>
<thead>
<tr>
<th>DB31, ... DBX12.2 - DBX12.3</th>
<th>2nd software limit switch plus or minus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st software limit switch for the plus and minus directions is effective.</td>
<td>2nd software limit switch for the plus and minus directions is not effective.</td>
</tr>
</tbody>
</table>

Signal state 0 or edge change 1 → 0

Corresponding to...

- MD36110 $MA_POS_LIMIT_PLUS (1st software limit switch plus)
- MD36130 $MA_POS_LIMIT_PLUS2 (2nd software limit switch plus)
- MD36100 $MA_POS_LIMIT_MINUS (1st software limit switch minus)
- MD36120 $MA_POS_LIMIT_MINUS2 (2nd software limit switch minus)

---

**19.2.4 Signals from axis/spindle (DB31, ...)**

<table>
<thead>
<tr>
<th>DB31, ... DBX60.2 - DBX60.3</th>
<th>Encoder limit frequency exceeded 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder limit frequency exceeded 2</td>
<td></td>
</tr>
</tbody>
</table>

Edge evaluation: No | Signal(s) updated: Cyclically

**Signal state 1**

The limit frequency set in the machine data:

- MD36300 $MA_ENC_FREQ_LIMIT (encoder limit frequency) has been exceeded.

The reference point for the position measuring system involved has been lost (IS: Referenced/synchronized has a signal state 0). Closed-loop position control is no longer possible. Spindles continue to run with closed-loop speed control. Axes are stopped with a fast stop (with open-circuit position control loop) along a speed setpoint ramp.

**Signal state 0**

The limit frequency set in machine data:

- MD36300 $MA_ENC_FREQ_LIMIT is no longer exceeded.

For the edge change 1 → 0, the encoder frequency must have fallen below the value of machine data:

- MD36302 $MA_ENC_FREQ_LIMIT_LOW (encoder limit frequency for encoder resynchronization)
### 19.3 Continuous-path mode, exact stop and LookAhead (B1)

#### 19.3.1 Signals from channel (DB21, ...)

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>DBX36.3</th>
<th>All axes stationary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong> no</td>
<td><strong>Signal(s) updated:</strong> cyclic</td>
<td></td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>All axes assigned to the channel are stationary with interpolator end. No other traversing movements are active.</td>
<td></td>
</tr>
</tbody>
</table>
### 19.3.2 Signals from axis/spindle (DB31, ...)

<table>
<thead>
<tr>
<th>DB31, ...</th>
<th>DBX60.6</th>
<th>Position reached with exact stop coarse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
<td></td>
</tr>
</tbody>
</table>

**Signal state 1 or edge change 0 → 1**
- The axis is in the appropriate exact stop and no interpolator is active for the axis and:
  - The control is in the reset state (reset key or end of program).
  - The axis was last programmed as a positioning axis or positioning spindle (initial setting of supplementary axis: Positioning axis).
  - The path motion was terminated with NC stop.
  - The spindle is in the closed-loop positioncontrolled mode ([SPCON/SPOS instruction](#)) and is stationary.
  - The axis is switched from closed-loop speedcontrolled to closed-loop positioncontrolled mode with IS "position measuring system".

**Signal state 0 or edge change 1 → 0**
- The axis is not in the appropriate exact stop or the interpolator is active for the axis or:
  - The path motion was terminated with NC stop.
  - The spindle is in the closed-loop speedcontrolled mode ([SPCOF/SPOSA instruction](#)).
  - The "followup" mode is active for the axis.
  - The "parking" mode is active for the axis.
  - The axis is switched from closed-loop positioncontrolled to closed-loop speedcontrolled mode with IS "position measuring system".

Signal irrelevant for ...
- Rotary axes that are programmed as rounding axes.

Corresponding to ...
- MD36000 `$MA_STOP_LIMIT_COARSE` (exact stop coarse)

---

<table>
<thead>
<tr>
<th>DB31, ...</th>
<th>DBX60.7</th>
<th>Position reached with exact stop fine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
<td></td>
</tr>
</tbody>
</table>

**Signal state 1 or edge change 0 → 1**
- Refer to DB31, ... DBX60.6 (position reached with exact stop coarse).

**Signal state 0 or edge change 1 → 0**
- Refer to DB31, ... DBX60.6 (position reached with exact stop coarse).

Signal irrelevant for ...
- Rotary axes that are programmed as rounding axes.

Corresponding to ...
- MD36010 `$MA_STOP_LIMIT_FINE` (exact stop fine)
19.4 Travel to fixed stop (F1)

19.4.1 Signals to axis/spindle (DB31, ...)

<table>
<thead>
<tr>
<th>DB31, ...</th>
<th>Acknowledge fixed stop reached</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX1.1</td>
<td></td>
</tr>
</tbody>
</table>

**Edge evaluation: no**  
**Signal(s) updated: cyclic**

**Signal state 1 or edge change 0 → 1**
Significance after the fixed stop has been reached: DB31, ... DBX62.5 (fixed stop reached) = 1
- The axis presses against the fixed stop with the clamping torque:
- The fixed stop monitoring window is activated.
- A block change is executed.

**Signal state 0**
Edge change 1 → 0
Significance after the fixed stop has been reached: DB31, ... DBX62.5 (fixed stop reached) = 1
- The axis presses against the fixed stop with the clamping torque.
- The fixed stop monitoring window is activated.
- A block change is not executed and the channel message "Wait: Auxiliary function acknowledgment missing" is displayed.
Meaning after the fixed stop has been reached:
IS "Fixed stop reached" DB31, ... DBX62.5 = 1
- The function is aborted, the alarm "20094 axis %1 Function aborted" is output.
Significance when de-selecting the function FXS=0 via the part program:
- The torque limiting and the monitoring of the fixed stop monitoring window are canceled.

**IS irrelevant for ...**
MD37060 $MA_FIXED_STOP_ACKN_MASK  
(monitoring PLC acknowledgments for travel to fixed stop)
= 0 or 1

**Corresponding to ...**
MD37060 $MA_FIXED_STOP_ACKN_MASK  
(monitoring PLC acknowledgments for travel to fixed stop)
DB31, ... DBX62.5 (fixed stop reached)

---

<table>
<thead>
<tr>
<th>DB31, ...</th>
<th>Sensor for fixed stop</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX1.2</td>
<td></td>
</tr>
</tbody>
</table>

**Edge evaluation: no**  
**Signal(s) updated: cyclic**

**Signal state 1 or edge change 0 → 1**
Fixed stop is reached.

**Signal state 0 or edge change 1 → 0**
Fixed stop is not reached.

**Corresponding to ...**
The signal is only active if:
MD37040 $MA_FIXED_STOP_BY_SENSOR = 1
Enable travel to fixed stop

<table>
<thead>
<tr>
<th>Signal state 1 or edge change</th>
<th>Meaning when \texttt{FXS} function is selected using the part program (IS &quot;Activate travel to fixed stop&quot; = 1):</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 → 1</td>
<td>Travel to fixed stop is enabled and the axis traverses from the start position at the programmed velocity to the programmed target position.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal state 0</th>
<th>Meaning when \texttt{FXS} function is selected using the part program (IS &quot;Activate travel to fixed stop&quot; = 1):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge change 1 → 0</td>
<td>The axis is stationary at the start position with reduced torque.</td>
</tr>
<tr>
<td></td>
<td>The channel message &quot;wait&quot;: Auxiliary function acknowledgment missing&quot; is displayed.</td>
</tr>
<tr>
<td></td>
<td>Before the fixed stop has been reached (IS &quot;fixed stop reached&quot; = 0):</td>
</tr>
<tr>
<td></td>
<td>→ Travel to fixed stop is aborted.</td>
</tr>
<tr>
<td></td>
<td>→ The alarm &quot;20094: Axis%1 Function aborted&quot; is displayed.</td>
</tr>
</tbody>
</table>

IS irrelevant for ...

Corresponding to ...

<table>
<thead>
<tr>
<th>MD37060 $MA_FIXED_STOP_ACKN_MASK (monitoring PLC acknowledgments for travel to fixed stop)</th>
</tr>
</thead>
<tbody>
<tr>
<td>= 0 or 2</td>
</tr>
</tbody>
</table>

Activate travel to fixed stop

<table>
<thead>
<tr>
<th>Signal state 1 or edge change</th>
<th>The &quot;Travel to fixed stop&quot; function is active. This signal is used for analog drives in order, for example, to activate the current or torque limitation parameterized in the actuator.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 → 1</td>
<td>The &quot;Travel to fixed stop function&quot; is not active.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal state 0 or edge change</th>
<th>The &quot;Travel to fixed stop function&quot; is not active.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 → 0</td>
<td>The &quot;Travel to fixed stop&quot; function is active. This signal is used for analog drives in order, for example, to activate the current or torque limitation parameterized in the actuator.</td>
</tr>
</tbody>
</table>

Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2
### Help function output to PLC (H2)

#### 19.5.1 Signals to channel (DB21, ...)

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>DBX30.5</th>
<th>Activate associated M01</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edge evaluation: no</td>
<td>Signal(s) updated:</td>
<td></td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>PLC signals the NCK that the associated \texttt{M01} (help function) should be activated.</td>
<td></td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>De-activate the associated \texttt{M01} (help function).</td>
<td></td>
</tr>
</tbody>
</table>

Corresponding to .... DB21, ... DBX 318.5 (associated \texttt{M01} active)

#### 19.5.2 Signals from channel (DB21, ...)

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>DBB58, DBB60 - DBB65</th>
<th>M, S, T, D, H, F functions Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: job-controlled by NCK</td>
<td></td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>One M, S, T, An D, H, or F function has been output to the interface with a new value together with the associated change signal at the beginning of an OB1 cycle. In this case, the change signal indicates that the appropriate value is valid.</td>
<td></td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>The change signals are reset by the PLC basic program at the start of the next OB1 cycle. The value of the data involved is not valid.</td>
<td></td>
</tr>
<tr>
<td>DB21, ... DBX59.0 - DBX59.4</td>
<td>M fct. 1-5 not decoded</td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------------</td>
<td></td>
</tr>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: job-controlled by NCK</td>
<td></td>
</tr>
</tbody>
</table>
| Signal state 1 or edge change 0 → 1 | M function is greater than 99 (for extended address = 0) or for extended address > 0, not included in the decoding list.  
This signal is available - together with the associated M change signal - for one OB1 cycle.  
Cause:  
- Incorrect M function programmed  
- M function not configured in the decoding list of the PLC  
Remedy, e.g.:  
- PLC sets read-in disable  
- Output of a PLC alarm |
| Signal state 0 or edge change 1 → 0 | M function less than 99 (for extended address = 0) or for extended address > 0 included in the decoding list. |

<table>
<thead>
<tr>
<th>DB21, ... DBB60 - DBB97</th>
<th>M, S, T, D, H, F functions Additional info &quot;Quick&quot; (fast acknowledgment)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: job-controlled by NCK</td>
</tr>
</tbody>
</table>
| Signal state 1 or edge change 0 → 1 | One M, S, T, An D, H, or F function has been output to the interface with a new value together with the associated change signal at the beginning of an OB1 cycle.  
In this case, the additional info "Quick" indicates the quick help function. |
| Signal state 0 or edge change 1 → 0 | The change signals are reset by the PLC basic program at the start of the next OB1 cycle.  
The value of the data involved is not valid. |

| DB21, ... DBB68 - DBB97 | M functions 1 to 5  
Extended address M functions 1 to 5 |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: job-controlled by NCK</td>
</tr>
</tbody>
</table>
| Signal state 1 or edge change 0 → 1 | Up to 5 M functions programmed in an NC block are simultaneously made available here as soon as the M change signals are available.  
Value range of M functions: 0 to 99999 9999; integer number  
Value range of the extended address: 0 to 99; integer number  
The M functions remain valid until they are overwritten by new M functions. |
| Signal state 0 or edge change 1 → 0 |  
- After the PLC has ramped-up.  
- All help functions are deleted before a new function is entered. |
| Application example(s)    | Decoding and evaluation of M functions that are not decoded as standard or via a list.  
Using the extended address, the M function can be assigned to another channel that does not correspond to that channel in which the program is running. |
| Special cases, errors, ...... | For M00 to M99 the extended address = 0. |
## S functions 1 to 3

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>DBB98 - DBB115</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Extended address S functions 1 to 3</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Edge evaluation:</strong> No</td>
<td><strong>Signal(s) updated:</strong> job-controlled by NCK</td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>Up to 3 S functions programmed in an NC block are simultaneously made available here as soon as the S change signals are available.</td>
</tr>
<tr>
<td></td>
<td>Value range of the spindle speed: 0 to 999 999; integer number</td>
</tr>
<tr>
<td></td>
<td>Value range of the extended address: 0 to 6; integer number</td>
</tr>
<tr>
<td></td>
<td>The S functions remain valid until they are overwritten by new S functions.</td>
</tr>
<tr>
<td><strong>Signal state 0 or edge change 1 → 0</strong></td>
<td>● After the PLC has ramped-up.</td>
</tr>
<tr>
<td></td>
<td>● All help functions are deleted before a new function is entered.</td>
</tr>
<tr>
<td><strong>Application example(s)</strong></td>
<td>Spindle speed control by the PLC.</td>
</tr>
<tr>
<td></td>
<td>The extended address is used to program for which spindle the S word is valid.</td>
</tr>
<tr>
<td></td>
<td>E.g.: S2=500</td>
</tr>
</tbody>
</table>

## T function 1

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>DBB118</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong> No</td>
<td><strong>Signal(s) updated:</strong> job-controlled by NCK</td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>The T function programmed in an NC block is made available here as soon as the T change signal is available.</td>
</tr>
<tr>
<td></td>
<td>Value range of T functions: 0 to 99 999 999; integer number</td>
</tr>
<tr>
<td></td>
<td>The T function remains valid until it is overwritten by a new T function.</td>
</tr>
<tr>
<td><strong>Signal state 0 or edge change 1 → 0</strong></td>
<td>● After the PLC has ramped-up.</td>
</tr>
<tr>
<td></td>
<td>● All help functions are deleted before a new function is entered.</td>
</tr>
<tr>
<td><strong>Application example(s)</strong></td>
<td>Control of automatic tool selection.</td>
</tr>
<tr>
<td><strong>Special cases, errors, ......</strong></td>
<td>With T0, the current tool is removed from the tool holder but not replaced by a new tool (default configuration of the machine manufacturer).</td>
</tr>
<tr>
<td><strong>Remark</strong></td>
<td>8 decade T numbers are only available as T function 1.</td>
</tr>
</tbody>
</table>

## D function 1

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>DBB129</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong> No</td>
<td><strong>Signal(s) updated:</strong> job-controlled by NCK</td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>The D function programmed in an NC block is made available here as soon as the D change signal is available.</td>
</tr>
<tr>
<td></td>
<td>Value range of D functions: 0 to 999; integer number</td>
</tr>
<tr>
<td></td>
<td>The D function remains valid until it is overwritten by a new D function.</td>
</tr>
<tr>
<td><strong>Signal state 0 or edge change 1 → 0</strong></td>
<td>● After the PLC has ramped-up.</td>
</tr>
<tr>
<td></td>
<td>● All help functions are deleted before a new function is entered.</td>
</tr>
<tr>
<td><strong>Application example(s)</strong></td>
<td>Implementation of protective functions.</td>
</tr>
<tr>
<td><strong>Special cases, errors, ......</strong></td>
<td>D0 is reserved for deselecting the current tool offset.</td>
</tr>
</tbody>
</table>
### Help function output to PLC (H2)

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>DBB140 - DBB157</th>
<th>H functions 1 to 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Extended address</strong></td>
<td>H functions 1 to 3</td>
<td></td>
</tr>
<tr>
<td><strong>Edge evaluation:</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>Signal(s) updated:</strong></td>
<td>job-controlled by NCK</td>
<td></td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>Up to 3 H functions programmed in an NC block are made available here simultaneously as soon as the H change signals are available.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value range of the H function: Floating point (corresponding to the MC5+ format)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value range of the extended address: 0 to 99; integer number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The H functions remain valid until they are overwritten by new H functions.</td>
<td></td>
</tr>
<tr>
<td><strong>Signal state 0 or edge change 1 → 0</strong></td>
<td>• After the PLC has ramped-up.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• All help functions are deleted before a new function is entered.</td>
<td></td>
</tr>
<tr>
<td><strong>Application example(s)</strong></td>
<td>Switching functions on the machine.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>DBB158 - DBB193</th>
<th>F functions 1 to 6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Extended address</strong></td>
<td>F functions 1 to 6</td>
<td></td>
</tr>
<tr>
<td><strong>Edge evaluation:</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>Signal(s) updated:</strong></td>
<td>job-controlled by NCK</td>
<td></td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>Up to 6 F functions (one path feed and up to 5 axis-specific feeds for positioning axes) are made available here simultaneously as soon as the F change signals are available.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value range of F function: Floating point (corresponding to the MC5+ format)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value range of the extended address: 0 to 18; integer number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The extended address of the F function is generated from the feed type (path feed or axis-specific feed) and the axis names.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>It is coded as follows:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Path feed; e.g.: F=1000 e.g.: F=1000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 to 18: Machine axis number of the positioning axis e.g.: FA[X1]=500</td>
<td></td>
</tr>
<tr>
<td></td>
<td>for an axis-specific feed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The F functions remain until they are overwritten by new F functions.</td>
<td></td>
</tr>
<tr>
<td><strong>Signal state 0 or edge change 1 → 0</strong></td>
<td>• After the PLC has ramped-up.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• All help functions are deleted before a new function is entered.</td>
<td></td>
</tr>
<tr>
<td><strong>Application example(s)</strong></td>
<td>Control of programmed F word by the PLC, e.g. through overwriting of the set feed rate override.</td>
<td></td>
</tr>
<tr>
<td><strong>Corresponding to ...</strong></td>
<td>MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of F functions)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>DBB194 - DBB206</th>
<th>Dynamic M functions: M0 - M99</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>Signal(s) updated:</strong></td>
<td>job-controlled by NCK</td>
<td></td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>The dynamic M signal bits are set by decoded M functions.</td>
<td></td>
</tr>
</tbody>
</table>
19.5 Help function output to PLC (H2)

**Dynamic M functions: M0 - M99**

| Signal state 0 or edge change 1 → 0 | For a general help function output, the dynamic M signal bits are acknowledged by the PLC basic program after the OB1 has been completely run-through (executed once). For a fast help function output, after the PLC identifies the help functions, they are acknowledged in the same OB40 cycle. |
| Application example(s) | Spindle clockwise/counterclockwise rotation, switch coolant ON/OFF |

**Associated M01/M00 active**

| Signal state 1 or edge change 0 → 1 | This bit indicates that an M00 or M01 help function is active if the appropriate associated M00 and M01 (help functions) were enabled/activated beforehand. |
| Signal state 0 or edge change 1 → 0 | No associated M00/M01 help functions active. |
| Corresponding to ... | DB21, ... DBX30.5 (activate associated M01) |

**Signals from axis/spindle (DB31, ...)**

**F auxiliary function for positioning axis**

| Edge evaluation: no | Signal(s) updated: Jobcontrolled |
| The values of the F help functions for positioning axes are stored here. The axis to which each value applies is determined by the extended address. |

**M auxiliary function for spindle**

| Edge evaluation: no | Signal(s) updated: Jobcontrolled |
| The values for the M3, M4, M5 help functions are sent to the associated interface for the addressed spindle. |

**S auxiliary function for spindle**

| Edge evaluation: no | Signal(s) updated: Jobcontrolled |
| The values for the S help functions are sent to the associated interface for the addressed spindle. |
19.6 Mode group, channel, program operation, reset response (K1)

19.6.1 Signals to mode group (DB11)

<table>
<thead>
<tr>
<th>DB11 DBX0.0</th>
<th>AUTOMATIC mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>AUTOMATIC mode is selected by the PLC program.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>AUTOMATIC mode is not selected by the PLC program.</td>
</tr>
<tr>
<td>Signal irrelevant for ...</td>
<td>DB11 DBX0.4 (operating mode, changeover inhibit) = 1</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB11 DBX6.0 (active AUTOMATIC mode)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB11 DBX0.1</th>
<th>MDA mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>MDA mode is selected by the PLC program.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>MDA mode is not selected by the PLC program.</td>
</tr>
<tr>
<td>Signal irrelevant for ...</td>
<td>DB11 DBX0.4 (operating mode, changeover inhibit) = 1</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB11 DBX6.1 (active MDA mode)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB11 DBX0.2</th>
<th>JOG mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>JOG mode is selected by the PLC program.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>JOG mode is not selected by the PLC program.</td>
</tr>
<tr>
<td>Signal irrelevant for ...</td>
<td>DB11 DBX0.4 (operating mode, changeover inhibit) = 1</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB11 DBX6.2 (active JOG mode)</td>
</tr>
</tbody>
</table>
### Mode change disable

**DB11 DBX0.4**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>The currently active mode (JOG, MDA or AUTOMATIC) of the mode group cannot be changed. The machine functions that can be selected within a mode group can be changed.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>The mode of the mode group can be changed.</td>
</tr>
</tbody>
</table>

### Mode group stop

**DB11 DBX0.5**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>An NC Stop is activated for all the channels of the mode group. The channel status of all the active channels changes to the channel status &quot;interrupted&quot;. All of the channels in channel status &quot;reset&quot; remain in the channel status &quot;reset&quot;. Programs that are running at this point are immediately interrupted (at the earliest possible point, even within a block) and the program status changes to &quot;stopped&quot;. All the moving axes of the mode group are decelerated according to their acceleration characteristics without contour violation. The program can be restarted with NC Start. None of the spindles of that mode group are affected.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Channel status and program execution are not influenced.</td>
</tr>
<tr>
<td>Special cases, errors, ... ...</td>
<td>All the axes of a mode group that are not triggered by a program or a program block (e.g. axes traverse because traverse keys are being pressed on the machine control panel) decelerate to rest with mode group stop.</td>
</tr>
</tbody>
</table>
### Mode group stop axes plus spindles

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>An NC Stop is activated for all the channels of the mode group. The channel status of all of the active channels changes to the channel status &quot;interrupted&quot;. All of the channels in channel status &quot;reset&quot; remain in the channel status &quot;reset&quot;. Programs that are running at this point are immediately interrupted (at the earliest possible point, even within a block) and the program status changes to &quot;stopped&quot;. All the moving axes and spindles of the mode group are decelerated according to their acceleration characteristics without contour violation. The program can be restarted with NC Start.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Channel status and program execution are not influenced.</td>
</tr>
</tbody>
</table>

### Mode group reset

<table>
<thead>
<tr>
<th>Edge evaluation: Yes</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>A reset is activated for all the channels of the mode group. All of the channels are then in the channel status &quot;reset&quot;. All of the current programs are then in the program status &quot;aborted&quot;. All moving axes and spindles are decelerated to zero speed according to their acceleration ramp without contour violation. The initial settings are set (e.g. for G functions). The alarms for the mode group are cleared if they are not POWER ON alarms.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Channel status and program execution are not influenced by this signal.</td>
</tr>
</tbody>
</table>

### Machine function TEACH IN

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>Machine function TEACH IN is activated in JOG mode for the mode group.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Machine function TEACH IN is not activated.</td>
</tr>
<tr>
<td>Signal irrelevant for ...</td>
<td>If JOG mode is not active.</td>
</tr>
</tbody>
</table>

### Special cases, errors, ......

- All the axes and spindles of a mode group that are not triggered by a program or a program block (e.g. axes traverse because traverse keys are pressed on the machine control panel, spindles are controlled by the PLC) decelerate to rest with "mode group stop plus spindles".
- An alarm that withdraws the interface signal DB11 DBX6.3 (mode group ready) ensures that all channels of the mode group are no longer in the reset status.
- In order to switch to another operating mode, a mode group reset (DB11 DBX0.7) must then be initiated.

### Additional references

- Operating Manual HMI (corresponding to the used software)
### Machine function REPOS

**DB11 DBX1.1**  
**Machine function REPOS**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>Machine function REPOS is activated in JOG mode for the mode group.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Machine function REPOS is not activated.</td>
</tr>
</tbody>
</table>

Signal irrelevant for... JOG mode is not active.

**Application example(s)**  
When a fault occurs when executing a part program (e.g. tool breakage), the axis is manually moved away from the fault location in the JOG mode in order to be able to replace the tool. The axis can then be manually returned to the exact previous position using the REPOS machine function so that the program can be continued in the automatic mode.

**Additional references**  
Operating Manual HMI (corresponding to the used software)

### Machine function REF

**DB11 DBX1.2**  
**Machine function REF**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>Machine function REF is activated in the JOG mode for the mode group.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Machine function REF is not activated.</td>
</tr>
</tbody>
</table>

Signal irrelevant for... If JOG mode is not active.

**Additional references**  
See Section "R1: Referencing (Page 1149)"

### Single block type B

**DB11 DBX1.6**  
**Single block type B**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated:</th>
</tr>
</thead>
</table>
| Signal state 1 or edge change 0 → 1 | Bit set and DB11 DBX1.7 not set: Response across modes  
- All channels are stopped.  
- All channels receive a start command.  
- Channel KS stops at the end of the block.  
- The channels KA receive a STOPATEND.  
(Comparable with DB21, ... DBX7.2 (NC Stop at the block limit).)  
- All channels are stopped at a block limit (at some point in time).  
(If DB11 DBX1.6 and DB11 DBX1.7 are set simultaneously, it is impossible to determine which single block type is required. The control then assumes: No single block across mode groups). |
| Signal state 0 or edge change 1 → 0 | If Bit DB11 DBX1.6 is not set and Bit DB11 DBX1.7 is set, then it is single block type A.  
(If DB11 DBX1.6 and DB11 DBX1.7 are not set, it is impossible to determine which single block type is required. The control then assumes: No single block across mode groups). |

**Corresponding to...** Single block type A
### 19.6 Mode group, channel, program operation, reset response (K1)

<table>
<thead>
<tr>
<th>DB11 DBX1.7</th>
<th>Single block type A</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong> No</td>
<td><strong>Signal(s) updated:</strong></td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>DB11 DBX1.7 set and DB11 DBX1.6 not set: Response across modes</td>
</tr>
<tr>
<td></td>
<td>• All channels are stopped.</td>
</tr>
<tr>
<td></td>
<td>• All channels receive a start (start key).</td>
</tr>
<tr>
<td></td>
<td>• Channel KS stops at the end of the block (due to single-block).</td>
</tr>
<tr>
<td></td>
<td>• Channels KA receive a STOP command (analogous to Stop key).</td>
</tr>
<tr>
<td></td>
<td>• All channels are stopped. (deceleration phase of all KAs).</td>
</tr>
<tr>
<td></td>
<td>(If DB11 DBX1.6 and DB11 DBX1.7 are set simultaneously, it is impossible to determine which single block type is required. The control then assumes: No single block across mode groups).</td>
</tr>
</tbody>
</table>

| Signal state 0 or edge change 1 → 0 | If DB11 DBX1.7 is not set and DB11 DBX1.6 is set, then it is single block type B. |
| | (If DB11 DBX1.6 and DB11 DBX1.7 are not set, it is impossible to determine which single block type is required. The control then assumes: No single block across mode groups). |

Corresponding to ... Single block type B

### 19.6.2 Signals from the mode group (DB11)

<table>
<thead>
<tr>
<th>DB11 DBX4.0</th>
<th>Selected mode AUTOMATIC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong> No</td>
<td><strong>Signal(s) updated:</strong> Cyclically</td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>AUTOMATIC mode is selected by HMI.</td>
</tr>
</tbody>
</table>

| Signal state 0 or edge change 1 → 0 | AUTOMATIC mode is not selected by HMI. |

<table>
<thead>
<tr>
<th>DB11 DBX4.1</th>
<th>Selected mode MDA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong></td>
<td><strong>Signal(s) updated:</strong> Cyclically</td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>MDA mode is selected by HMI.</td>
</tr>
</tbody>
</table>

| Signal state 0 or edge change 1 → 0 | MDA mode is not selected by HMI. |
### DB11

#### DBX4.2

**Data block**

<table>
<thead>
<tr>
<th><strong>Selected JOG mode</strong></th>
<th><strong>Signal(s) from BAG (HMI → PLC)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or</td>
<td>JOG mode is selected by HMI.</td>
</tr>
<tr>
<td>edge change 0 → 1</td>
<td></td>
</tr>
<tr>
<td>Signal state 0 or</td>
<td>JOG mode is not selected by HMI.</td>
</tr>
<tr>
<td>edge change 1 → 0</td>
<td></td>
</tr>
</tbody>
</table>

#### DBX5.0

**Selected machine function TEACH IN**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or</td>
<td>The machine function TEACH IN is selected by HMI within BAG.</td>
</tr>
<tr>
<td>edge change 0 → 1</td>
<td></td>
</tr>
<tr>
<td>Signal state 0 or</td>
<td>The machine function TEACH IN is not selected by HMI.</td>
</tr>
<tr>
<td>edge change 1 → 0</td>
<td></td>
</tr>
</tbody>
</table>

**Additional references**

Operating Manual HMI (corresponding to the used software)

#### DBX5.1

**Selected REPOS machine function**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or</td>
<td>The machine function REPOS is selected by HMI within BAG.</td>
</tr>
<tr>
<td>edge change 0 → 1</td>
<td></td>
</tr>
<tr>
<td>Signal state 0 or</td>
<td>The machine function REPOS is not selected by HMI.</td>
</tr>
<tr>
<td>edge change 1 → 0</td>
<td></td>
</tr>
</tbody>
</table>

**Application example(s)**

When a fault occurs when executing a part program (e.g. tool breakage), the axis is manually moved away from the fault location in the JOG mode in order to be able to replace the tool. The axis can then be manually returned to the exact previous position using the REPOS machine function so that the program can be continued in the automatic mode.

**Additional references**

Operating Manual HMI (corresponding to the used software)

#### DBX5.2

**Selected machine function REF**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or</td>
<td>The machine function REF is selected by HMI within BAG.</td>
</tr>
<tr>
<td>edge change 0 → 1</td>
<td></td>
</tr>
</tbody>
</table>
### 19.6 Mode group, channel, program operation, reset response (K1)

<table>
<thead>
<tr>
<th>DB11</th>
<th>DBX5.2</th>
<th>Selected machine function REF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Signal state 0 or edge change 1 → 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The machine function REF is not selected by HMI.</td>
</tr>
</tbody>
</table>

Additional references: See Section "R1: Referencing (Page 1149)"

<table>
<thead>
<tr>
<th>DB11</th>
<th>DBX6.0</th>
<th>Active mode AUTOMATIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data block</td>
<td>Signal(s) from the mode group (NCK → PLC)</td>
<td></td>
</tr>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
<td></td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>AUTOMATIC mode is active.</td>
<td></td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>AUTOMATIC mode is not active.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB11</th>
<th>DBX6.1</th>
<th>Active mode MDA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Edge evaluation:</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>MDA mode is active.</td>
<td></td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>MDA mode is not active.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB11</th>
<th>DBX6.2</th>
<th>Active JOG mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>JOG mode is active.</td>
<td></td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>JOG mode is not active.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB11</th>
<th>DBX6.3</th>
<th>Mode group ready</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>This signal is set after power on and all of the voltage have been established. The mode group is now ready and part programs can be executed and axes traversed in the individual channels.</td>
<td></td>
</tr>
</tbody>
</table>
### Mode group ready

<table>
<thead>
<tr>
<th>DB11 DBX6.3</th>
<th>Mode group ready</th>
</tr>
</thead>
</table>
| Signal state 0 or edge change 1 → 0 | The mode group is not ready. Possible causes for this are:  
  - A critical axis or spindle alarm is present  
  - Hardware faults  
  - The mode group has been incorrectly configured (machine data)  
If the mode group ready changes to signal state "0", then:  
  - The axis and spindle drives are braked down to standstill with the max. braking current.  
  - The signals from the PLC to the NCk are brought into an inactive state (cleared state). |

#### Special cases, errors, ......  
An alarm that withdraws the interface signal DB11 DBX6.3 (mode group ready) ensures that all channels of the mode group are no longer in the reset state. In order to switch to another operating mode, a mode group reset (DB11 DBX0.7) must be made.

### All channels in the reset state

<table>
<thead>
<tr>
<th>DB11 DBX6.7</th>
<th>All channels in the reset state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>All the channels that belong to this mode group are in the &quot;channel status reset&quot; state (DB21, ... DBX7.7).</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>At least one of the channels in the mode group is not in &quot;channel status reset&quot; (DB21, ... DBX7.7).</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB21, ... DBX7.7 (channel state, reset)</td>
</tr>
</tbody>
</table>

### Active machine function TEACH IN

<table>
<thead>
<tr>
<th>DB11 DBX7.0</th>
<th>Active machine function TEACH IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>Machine function TEACH IN is active in the mode group.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Machine function TEACH IN is not active.</td>
</tr>
<tr>
<td>Additional references</td>
<td>Operating Manual HMI (corresponding to the used software)</td>
</tr>
</tbody>
</table>

### Active REPOS machine function

<table>
<thead>
<tr>
<th>DB11 DBX7.1</th>
<th>Active REPOS machine function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>Machine function REPOS is active in the mode group.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Machine function REPOS is not active.</td>
</tr>
</tbody>
</table>
Active REPOS machine function

Application example(s)
- When a fault occurs when executing a part program (e.g. tool breakage), the axis is manually moved away from the fault location in the JOG mode in order to be able to replace the tool.
- The axis can then be manually returned to the exact previous position using the REPOS machine function so that the program can be continued in the automatic mode.

Additional references
- Operating Manual HMI (corresponding to the used software)

Active machine function REF

Edge evaluation: No
- Signal(s) updated: Cyclically
- Machine function REF is active in the mode group.
- Machine function REF is not active.

Additional references
- See Section "R1: Referencing (Page 1149)"

19.6.3 Signals to channel (DB21, ...)

Activate single block
- In AUTOMATIC and MDI modes, the operator must enable processing of each individual part program block of the part program selected in the channel by reactivating NC Start.

Special cases, errors, .......
- In the case of active tool offset, intermediate blocks are inserted, when necessary. These blocks must also be enabled using NC Start.
- In a series of G33 blocks, a single block is only operative if "dry run feed" is selected.
- In the case of a decoding single block, calculation blocks are not processed in the single step.

Activate M01
- Activation of program control "Conditional stop" M01 is requested.
- Activation of program control "Conditional stop" M01 is not requested.

Corresponding to ...
- DB21, ... DBX35.3 (program status interrupted)
- DB21, ... DBX24.5 (M01 selected)
- DB21, ... DBX32.5 (M0/M01 active)
### DB21, ... DBX1.6

<table>
<thead>
<tr>
<th>PLC action completed</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: no</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>

At the end of the block search, concluding action blocks are executed:
- DB21, ... DBX32.3 (action block active) == 1 AND
- DB21, ... DBX32.6 (last action block active) == 1

Alarm "10208 Channel <Channel Number> Issue NC Start to continue program" notifies that NC Start must be reactivated to resume the part program starting from the target block.

If other actions are to be executed by the PLC user program prior to the NC Start (e.g. tool change), Search mode can be parameterized as follows:
- MD11450 $MN_SEARCH_RUN_MODE = 1
- Output of alarm delayed until the existing signal is reset.

| Signal state 1 | PLC action is completed. |
| Signal state 0 | PLC action is not yet completed. |

Corresponding to:
- DB21, ... DBX32.3 (action block active)
- DB21, ... DBX32.6 (last action block active)
- DB21, ... DBX33.4 (block search active)

### DB21, ... DBX1.7

<table>
<thead>
<tr>
<th>Activate program test</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: no</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>

Activation of the program test is requested.
During the program test, all motion commands of axes (not spindles) take place under "Axis disable."

**Notice!**
Due to the axis disable, the assignment of a tool magazine is not changed for the program test. The user / machine manufacturer must utilize a suitable PLC user program to ensure that the NC-internal tool management and the actual assignment of the tool magazine remain consistent. Refer to the program example included in the PLC Toolbox.

| Signal state 1 | Activation of the program test is requested. |
| Signal state 0 | Activation of the program test is not requested. |

Corresponding to:
- DB21, ... DBX25.7 (program test selected)
- DB21, ... DBX33.7 (program test active)

### DB21, ... DBB2

<table>
<thead>
<tr>
<th>Activate skip block</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>

Skip blocks marked in the part program with a slash (/) are not processed. If there is a series of skip blocks, the signal is only active if it is present before the first skip block of the series is decoded.

**Note**
The signal should be available prior to the start of the part program.

| Signal state 1 | Skip blocks marked in the part program with a slash (/) are processed. |
| Signal state 0 | Skip blocks marked in the part program with an slash (/) are processed. |

Corresponding to:
- DB21, ... DBX26.0 (skip block selected)
- DB21, ... DBX35.2 (program status stopped)
### Z1: NC/PLC interface signals

19.6 Mode group, channel, program operation, reset response (K1)

<table>
<thead>
<tr>
<th>DB21, ... DBX6.1</th>
<th>Read-in disable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong> No</td>
<td><strong>Signal(s) updated:</strong> Cyclically</td>
</tr>
</tbody>
</table>
| **Signal state 1** | The main run does not read in any preprocessed part program blocks.  
**Note** | The signal is only active in AUTOMATIC and MDI modes. |
| **Signal state 0** | The main run reads in preprocessed part program blocks. |
| **Corresponding to ...** | DB21, ... DBX35.0 (program status running) |

<table>
<thead>
<tr>
<th>DB21, ... DBX6.4</th>
<th>Program level abort</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong> Yes</td>
<td><strong>Signal(s) updated:</strong> Cyclically</td>
</tr>
<tr>
<td><strong>Edge change 0 → 1</strong></td>
<td>At each edge change 0 → 1 the current program level being processed (subprogram level, ASUB level, save routine) is immediately aborted. Processing of the part program continues at the next higher program level from the exit point.</td>
</tr>
<tr>
<td><strong>Edge change 1 → 0</strong></td>
<td>No effect.</td>
</tr>
<tr>
<td><strong>Special cases, errors, ......</strong></td>
<td>The main program level cannot be aborted with this IS, only with IS &quot;Reset&quot;.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB21, ... DBX7.0</th>
<th>NC Start disable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong> No</td>
<td><strong>Signal(s) updated:</strong> Cyclically</td>
</tr>
<tr>
<td><strong>Signal state 1</strong></td>
<td>The NC Start disable prevents a part program from being started with NC START signal DB21, ... DBX7.1 (NC Start) == 1.</td>
</tr>
<tr>
<td><strong>Signal state 0</strong></td>
<td>NC Start disable is not active.</td>
</tr>
<tr>
<td><strong>Special cases, errors, ......</strong></td>
<td>The start of a part program selected in the channel by part program command START in another channel (program coordination) is not prevented by the interface signal: DB21, ... DBX7.0 (NC Start disable) == 1.</td>
</tr>
<tr>
<td><strong>Corresponding to ...</strong></td>
<td>DB21, ... DBX7.1 (NC Start)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB21, ... DBX7.1</th>
<th>NC Start</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong> Yes</td>
<td><strong>Signal(s) updated:</strong> Cyclically</td>
</tr>
</tbody>
</table>
| **Edge change 0 → 1** | AUTOMATIC mode: The selected NC program is started or continued, or the auxiliary functions that were saved during the program interruption are output.  
If data is transferred from the PLC to the NC during program status "Program interrupted," then this data is immediately cleared at NC Start.  
MDI:mode  
The entered block information or part program blocks are released for execution. |
| **Edge change 1 → 0** | No effect. |
| **Corresponding to ...** | DB21, ... DBX7.0 (NC Start disable) |
### DB21, ... DBX7.2
**NC Stop at block limit**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>The current NC program is stopped after the current part program block has been completely processed. Otherwise, as for DB21, ... DBX7.3 (NC Stop).</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>No effect.</td>
</tr>
</tbody>
</table>

**Corresponding to**
- DB21, ... DBX7.3 (NC Stop)
- DB21, ... DBX7.4 (NC Stop axes plus spindles)
- DB21, ... DBX35.2 (program status stopped)
- DB21, ... DBX35.6 (channel status interrupted)

### DB21, ... DBX7.3
**NC Stop**

<table>
<thead>
<tr>
<th>Edge evaluation: no</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>AUTOMATIC or MDI mode: Processing of the part program active in the channel is stopped. The axes (not spindles) are brought to a standstill within the assigned acceleration parameters.</td>
</tr>
<tr>
<td></td>
<td>- Program status: Stopped</td>
</tr>
<tr>
<td></td>
<td>- Channel status: Interrupted</td>
</tr>
<tr>
<td></td>
<td>JOG mode:</td>
</tr>
<tr>
<td></td>
<td>In JOG mode, incompletely traversed incremental paths (INC...) are retracted at the next NC Start.</td>
</tr>
<tr>
<td></td>
<td>Note</td>
</tr>
<tr>
<td></td>
<td>The signal must be present for at least one PLC cycle (OB1).</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>No effect.</td>
</tr>
<tr>
<td>Signal irrelevant for ...</td>
<td></td>
</tr>
<tr>
<td>Special cases, errors, ......</td>
<td></td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td></td>
</tr>
</tbody>
</table>
- DB21, ... DBX7.2 (NC Stop at block limit)
- DB21, ... DBX7.4 (NC Stop axes plus spindles)
- DB21, ... DBX35.2 (program status stopped)
- DB21, ... DBX35.6 (channel status interrupted)

### DB21, ... DBX7.4
**NC Stop axes plus spindles**

<table>
<thead>
<tr>
<th>Edge evaluation: no</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>See DB21, ... DBX7.3 (NC Stop). In addition, the spindles of the channel are stopped.</td>
<td></td>
</tr>
</tbody>
</table>
**19.6 Mode group, channel, program operation, reset response (K1)**

### DB21, ... DBX7.7

**Reset**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>The channel is reset. The initial settings are set (e.g. G functions). The alarms for the channel are cleared if they are not POWER ON alarms. The reset signal must be issued by the PLC (e.g. using a logic operation with the reset key on the MCP). The signal is only evaluated by the selected channel. The program status changes to &quot;Aborted&quot;, and the channel status changes to &quot;Channel status reset&quot;.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>No effect.</td>
</tr>
</tbody>
</table>

**Corresponding to ...**
- DB11, ... DBX0.7 (mode group reset)
- DB21, ... DBX35.7 (channel status reset)

### DB21, ... DBX31.0 - DBX31.2

**REPOS mode (A, B, C)**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meaning</td>
<td>REPOS mode for repositioning to the contour after an interruption of the traversal block</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mode</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>No REPOS mode active</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RMB: Repositioning to start of block</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>RMI: Repositioning to interrupt point</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RME: Repositioning to end of block</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RMN: Repositioning to the nearest path point</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Corresponding to ...**
- DB21, ... DBX25.0-2 (REPOS mode)
- DB31, ... DBX10.0 (REPOS_DELAY)

### DB21, ... DBX31.4

**REPOS mode change**

<table>
<thead>
<tr>
<th>Edge evaluation: Yes</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge change 0 → 1</td>
<td>REPOS mode has changed.</td>
</tr>
<tr>
<td>Edge change 1 → 0</td>
<td>REPOS mode has not changed.</td>
</tr>
</tbody>
</table>

**Corresponding to ...**
- DB21, ... DBX31.0-2 (REPOS mode)
- DB21, ... DBX319.0 (acknowledgement of REPOS mode change)

### 19.6.4 Signals from channel (DB21, ...)

<table>
<thead>
<tr>
<th>DB21, ... DBX32.3</th>
<th>Action block active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>The action block is being executed.</td>
</tr>
</tbody>
</table>
### Z1: NC/PLC interface signals

**19.6 Mode group, channel, program operation, reset response (K1)**

<table>
<thead>
<tr>
<th>DB21, ... DBX32.3</th>
<th><strong>Action block active</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 0</td>
<td>No action block active.</td>
</tr>
<tr>
<td>Additional references</td>
<td>Operating Manual HMI (corresponding to the used software)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB21, ... DBX32.4</th>
<th><strong>Approach block active</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>The approach block for the progress of the program with &quot;Block search with computation on contour&quot; is active, as with &quot;Block search with computation on block end point&quot; no approach block is created of its own. The axes are automatically positioned on the collected search position if ASUB exits with REPOSA during &quot;Block search with computation on contour&quot;.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>The search target is found during &quot;Block search with computation on contour&quot;.</td>
</tr>
<tr>
<td>Additional references</td>
<td>Programming Manual, Job Planning</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB21, ... DBX32.5</th>
<th><strong>M00/M01 active</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>
| Signal state 1 | The part program block is processed, the auxiliary functions are output, and:  
  - M00 is in the RAM  
  - M01 is in the RAM and IS "Activate M01" is active  
  The program status changes to "Stopped". |
| Signal state 0 |  
  - With DB21, ... DBX7.1 (NC Start)  
  - For a program abort as a result of a reset |
### 19.6 Mode group, channel, program operation, reset response (K1)

<table>
<thead>
<tr>
<th>DB21, ... DBX32.5</th>
<th>M00/M01 active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td></td>
</tr>
</tbody>
</table>

1. Data transfer to working memory
2. Block processed
3. NC block with M00
4. M change signal (1 PLC cycle time)
5. IS "M00/M01 active"
6. IS "Channel state active" (even when axes are moved in JOG mode)

**Corresponding to**
- DB21, ... DBX0.5 (activate M01)
- DB21, ... DBX24.5 (M01 selected)

### DB21, ... DBX32.6

<table>
<thead>
<tr>
<th>Last action block active</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edge evaluation:</strong> No</td>
</tr>
<tr>
<td><strong>Signal(s) updated:</strong> Cyclically</td>
</tr>
</tbody>
</table>

**Signal state 1**
The last action block is being executed. This means that all the action blocks on the NC side have been processed and the actions on the PLC side (ASUB, FC) or the operator such as overstore, mode change according to JOG/REPOS are possible. In this way the PLC, for example, can still perform a tool change before the start of movement.

**Signal state 0**
The last action block is not being executed. Action blocks contain the actions collected during "Block search with computation" such as
- Output help function H, M00, M01, M..
- Tool programming T, D, DL
- Spindle programming S-Value, M3/M4/M5/M19, SPOS
- Feed programming, F

**Additional references**
See Section "K1: Mode group, channel, program operation, reset response (Page 455)"

---

Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2

1615
### Block search active

| Signal state 1 | The block search function is active. It was selected from the operator interface and started using the interface signal: DB21, ... DBX7.1 (NC Start) |

| Signal state 0 | Search target found. |

**Application example(s):**
The block search function makes it possible to jump to a certain block within a part program and to start processing the part program from this block.

**Additional references:**
See Section "K1: Mode group, channel, program operation, reset response (Page 455)"

### M02/M30 active

| Signal state 1 | NC block with M02 or M30 (or M17 if a subprogram was started) is completely processed; if traversing motions are also programmed in this block, the signal is only output when the target position is reached.  
|                | The program was interrupted as a result of a reset, and the program status changes to "Aborted".  
|                | When MDI mode or machine functions REF or PRESET are selected  
|                | According to DB10 DBX56.2 (acknowledge emergency stop) |

| Signal state 0 | No program end or program abort  
|                | Status after activation of control  
|                | Start of an NC Program |

**Screen**

1. Data transfer to working memory  
2. Block processed  
3. NC block with M02  
4. M change signal (1 PLC cycle time)  
5. IS "M02/M30 active"
### 19.6 Mode group, channel, program operation, reset response (K1)

<table>
<thead>
<tr>
<th>DB21, ... DBX33.5</th>
<th>M02/M30 active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application example(s)</td>
<td>The PLC can detect the end of program processing with this signal and react appropriately.</td>
</tr>
</tbody>
</table>
| Special cases, errors, ...... | - The M02 and M30 functions have equal priority.  
- The interface signal: DB21, ... DBX33.5 (M02/M30 active) is available as steady-state signal after the end of the program.  
- Not suitable for automatic follow-on functions such as workpiece counting, bar feed, etc. For these functions, M02/M30 must be written in a separate block and the word M02/M30 or the decoded M signal used.  
- Auxiliary functions that could result in a read-in operation being stopped and any S values that are to be operative beyond M02/M30 must not be written in the last block of a program. |

<table>
<thead>
<tr>
<th>DB21, ... DBX33.6</th>
<th>Transformation active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>The NC command TRAORI (activate transformation) is programmed in the NC part program. This block has been processed by the NC and the transformation is now activated.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>No transformation active.</td>
</tr>
<tr>
<td>Additional references</td>
<td>Programming Manual, Job Planning</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB21, ... DBX33.7</th>
<th>Program test active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>Program control program test is active. Axis disable is set internally for all axes (not spindles). Therefore the machine axes do not move when a part program block or a part program is being processed. The axis movements are simulated on the user interface with changing axis position values. The axis position values for the display are generated from the calculated setpoints. The part program is processed in the normal way.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>Program control &quot;Program test&quot; is not active.</td>
</tr>
</tbody>
</table>
| Corresponding to ... | DB21, ... DBX1.7 (activate program test)  
DB21, ... DBX25.7 (program test selected) |

<table>
<thead>
<tr>
<th>DB21, ... DBX35.0</th>
<th>Program status running</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>
| Signal state 1 | The part program was started with the interface signal: DB21, ... DBX7.1 (NC Start) and is running.  
The running program was stopped with the interface signal: DB21, ... DBX6.1 (read-in disable) |
### Program status running

<table>
<thead>
<tr>
<th>Signal state 0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Program stopped by M00/M01 or NC Stop or operating mode change.</td>
<td></td>
</tr>
<tr>
<td>● If single-block mode, then the block is processed.</td>
<td></td>
</tr>
<tr>
<td>● End of program reached (M02/M30).</td>
<td></td>
</tr>
<tr>
<td>● Program aborted due to a reset.</td>
<td></td>
</tr>
<tr>
<td>● No actual block in the memory (e.g. for MDI).</td>
<td></td>
</tr>
<tr>
<td>● The actual block cannot be executed.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal irrelevant for ...</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The part program was started with the interface signal: DB21, ... DBX7.1 (NC Start) and is running.</td>
<td></td>
</tr>
</tbody>
</table>

### Special cases, errors, ......

The interface signal: DB21, ... DBX35.0 (program status running) does not change to 0 if workpiece machining is stopped due to the following events:

- A feed disable or spindle disable was output
- DB21, ... DBX6.1 (read-in disable)
- Feed correction to 0%
- The spindle and axis monitoring functions respond
- Position setpoints are entered in the NC program for axes in "follow-up mode," for axes without "servo enable," or for "parking axes"

### Program status wait

<table>
<thead>
<tr>
<th>Signal state 1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The running program is waiting for the program command WAIT_M or WAIT_E in an NC block. The wait condition specified in the WAIT command for the channel or channels has not yet been fulfilled.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal state 0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program status wait is not active.</td>
<td></td>
</tr>
</tbody>
</table>

### Program status stopped

<table>
<thead>
<tr>
<th>Signal state 1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The NC part program has been stopped by:</td>
<td></td>
</tr>
<tr>
<td>● DB21, ... DBX7.3 (NC Stop)</td>
<td></td>
</tr>
<tr>
<td>● DB21, ... DBX7.4 (NC Stop axes plus spindles)</td>
<td></td>
</tr>
<tr>
<td>● DB21, ... DBX7.2 (NC Stop at the block limit)</td>
<td></td>
</tr>
<tr>
<td>● Programmed M00 or M01 or Single-block mode</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal state 0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Program status stopped&quot; is not present.</td>
<td></td>
</tr>
</tbody>
</table>

### Corresponding to ...

Corresponding to DB21, ... DBX6.1 (read-in disable)
### Program status interrupted

<table>
<thead>
<tr>
<th>DB21, ... DBX35.3</th>
<th>Program status interrupted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>When the operating mode changes from AUTOMATIC or MDI (in stopped program status) to JOG, the program status changes to &quot;Interrupted&quot;. The program can be continued at the point of interruption in AUTOMATIC or MDI mode when NC Start is issued.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>&quot;Program status interrupted&quot; not available.</td>
</tr>
<tr>
<td>Special cases, errors, ......</td>
<td>The interface signal: DB21, ... DBX35.3 (program status interrupted) indicates that the part program can continue to be processed by restarting it.</td>
</tr>
</tbody>
</table>

### Program status aborted

<table>
<thead>
<tr>
<th>DB21, ... DBX35.4</th>
<th>Program status aborted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>The program has been selected but not started, or the current program was aborted with a reset.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>Program status interrupted is not active.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB21, ... DBX7.7 (reset)</td>
</tr>
</tbody>
</table>

### Channel status active

<table>
<thead>
<tr>
<th>DB21, ... DBX35.5</th>
<th>Channel status active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>
| Signal state 1 | In this channel:  
- A part program is presently being executed in Automatic or MDI mode  
- At least one axis is being traversed in JOG mode. |
| Signal state 0 | DB21, ... DBX35.3 (channel status interrupted) or DB21, ... DB35.7 (channel status reset) is present. |

### Channel status interrupted

<table>
<thead>
<tr>
<th>DB21, ... DBX35.6</th>
<th>Channel status interrupted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>
| Signal state 1 | The NC part program in AUTOMATIC or MDI mode or a traversing motion in JOG mode can be interrupted by:  
- DB21, ... DBX7.3 (NC Stop)  
- DB21, ... DBX7.4 (NC Stop axes plus spindles)  
- DB21, ... DBX7.2 (NC Stop at the block limit)  
- Programmed M00 or M01  
- Single-block mode  
After an NC Start the part program or the interrupted traversing movement can be continued. |
| Signal state 0 | DB21, ... DBX35.5 (channel status active) or DB21, ... DB35.7 (channel status reset) is present. |
### Channel status reset

| Signal state 1 | The signal changes to 1 as soon as the channel goes into the reset state, i.e. no processing taking place. |
| Signal state 0 | The signal is set to 0 as soon as processing takes place in the channel, e.g.:  
  - Execution of a part program  
  - Block search  
  - TEACH IN active  
  - Overstore active |

### Interrupt processing active

| Signal state 1 | One or more channels in the mode group are not in the desired operating mode as the result of an active interrupt routine. |
| Signal state 0 | All channels are operating in the required mode. |

**Note:** The signal is not set if an interrupt routine is running in a program mode.

### Channel is ready

| Signal state 1 | A channel is ready for a part program processing of machine axes, geometry axes and positioning axes. These are already allocated corresponding to machine configuration and the current program status of the concerned channels. |
| Signal state 0 | The concerned channel is not ready for a part program processing of machine axes, geometry axes and positioning axes. |

### Read-in disable is ignored

The following machine data is used to specify that the read-in disable (DB21, ... DBX6.1) is to be ignored:

- MD11602 $MN_ASUP_START_MASK, Bit 2 = 1 (start also permitted if read-in disable is active)
- MD20116 $MC_IGNORE_INHIBIT_ASUP (execute interrupt program in spite of read-in disable)
- MD20107 $MC_PROG_EVENT_IGN_INHIBIT (Prog events ignore read-in disable)

Part program blocks for which read-in disable is ignored are designated as read-in disable-inoperative.

### Read-in disable is active

Read-in disable is active (DB21, ... DBX6.1==1) AND part program block is read-in disable-inoperative.
**DB21, ... DBX37.6**

<table>
<thead>
<tr>
<th>Signal state 0</th>
<th>Read-in disable is not active (DB21, ... DBB6.1 == 0) OR (read-in disable is active (DB21, ... DBX6.1 == 1) AND part program block is read-in disable-operative)</th>
</tr>
</thead>
</table>

Corresponding to ...

DB21, ... DBX37.7 (Stop at block end is ignored during single block (SBL))

---

**DB21, ... DBX37.7**

<table>
<thead>
<tr>
<th>Edge evaluation:</th>
<th>Stop at block end is ignored during single block (SBL)</th>
</tr>
</thead>
</table>

Signal state 1

Single block is active (DB21, ... DBX0.4==1) AND part program block is single block-inoperative.

Signal state 0

Single block is not active (DB21, ... DBB0.4 == 0) OR (single block is active (DB21, ... DBX0.4 == 1) AND part program block is single block-operative)

Corresponding to ...

Read-in disable is ignored. DB21, ... DBX37.6 (read-in disable is ignored)

---

**DB21, ... DBB208 - DBB271**

<table>
<thead>
<tr>
<th>Active G function of groups 1 to 60</th>
</tr>
</thead>
</table>

**Edge evaluation:** No

**Signal state <> 0**

The G function displayed in BCD format or its mnemonic identifier is active in the G group.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1st G function of the G group</td>
</tr>
<tr>
<td>2</td>
<td>2nd G function of the G group</td>
</tr>
<tr>
<td>n</td>
<td>nth G function of the G group</td>
</tr>
</tbody>
</table>

**Signal state = 0**

No G function or G group mnemonic identifier is active.

**Application example(s)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Example:

**G90**

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Special cases, errors, ......**

In contrast to auxiliary functions, G functions are not output to the PLC subject to acknowledgement, i.e. processing of the part program is continued immediately after the G function output.

**Additional references**

A complete list of the G groups and G functions can be found in:

**References:** Programming Manual, Fundamentals.
### DB21, ... DBB317.1

**Workpiece setpoint reached**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state = 1</td>
<td>The number of machined workpieces (actual workpiece total) is <strong>equal</strong> to the number of workpieces to be machined (required number of workpieces): $AC_{ACTUAL_PARTS} == AC_{REQUIRED_PARTS}$</td>
</tr>
<tr>
<td>Signal state = 0</td>
<td>The number of machined workpieces (actual workpiece total) is <strong>not equal</strong> to the number of workpieces to be machined (required number of workpieces): $AC_{ACTUAL_PARTS} &lt;&gt; AC_{REQUIRED_PARTS}$</td>
</tr>
</tbody>
</table>

Corresponding to ... MD27880 $MC\_PART\_COUNTER$ (activation of workpiece counters)

Additional references Function Manual, Fundamentals, Section "K1: Mode group, channel, program operation, reset response" > "Program runtime / workpiece counter" > "Workpiece counter"

### DB21, ... DBX318.0

**ASUB is stopped**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>The signal is set to 1 if the control stops automatically prior to the end of the ASUB. The IS DB21, ... DBX318.0 (ASUB is stopped) is only supplied in the case &quot;Interrupt in a program mode and channel status stopped&quot;.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>The IS DB21, ... DBX318.0 (ASUB is stopped) is set to 0 with start and reset.</td>
</tr>
</tbody>
</table>
ASUB with REPOSA
is triggered in the sta‐
tus AUTOMATIC
mode stopped

If the interrupt routine is ended with REPOSA, then the following sequence is typical:
- NC Stop or an alarm is used to stop the part program.
- The control assumes program status "Stopped".
- The PLC initiates an ASUB via block FC9.
- Before the re-approach to the contour, the control stops and goes to program state "Stopped". IS DB21, ... DBX318.0 (ASUB is stopped) is set.
- The user presses Start. The IS DB21, ... DBX318.0 (ASUB is stopped) is reset, the re-approach motion is started.
- At the end of the re-approach motion, the FC9 signal "ASUB done" is set and the path of the interrupted part program is continued.

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>ASUB is stopped</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX318.0</td>
<td></td>
</tr>
</tbody>
</table>

Typical sequence of an ASUB with REPOSA:

- ASUB with REPOSA
  Event that brings the control into the "stopped" status occurs.
  Example: Stop key

- Control system stopped, ASUB is then started by the FC.

- Path motion of an ASUB with REPOSA

- Control has stopped automatically.

- IS "ASUB stopped" is set.

- FC9 block signals "ASUB Done"

- Path of the parts program

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>Block search via program test is active (SERUPRO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX318.1</td>
<td></td>
</tr>
</tbody>
</table>

**Edge evaluation:** No

**Signal(s) updated:** Cyclically

<table>
<thead>
<tr>
<th>Signal state 1</th>
<th>During the processing of the part program block in context of block search (internal channel status: &quot;Program test&quot;), up to the change of the target block in the main execution (Program status: &quot;Stopped&quot;).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 0</td>
<td>With the change of the target block in the main execution (internal channel status: &quot;Program test&quot; is deselected; stop condition: &quot;Search target found&quot; is displayed).</td>
</tr>
</tbody>
</table>

**Special cases, errors, ......**

The block search (SERUPRO) can only be activated in AUTOMATIC mode in program status "Abor‐
ted".
DB21, ... DBX319.0
Acknowledgement of the REPOS mode change

<table>
<thead>
<tr>
<th>Edge evaluation: Yes</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge change 0 → 1</td>
<td>The interface signal detected by the NC: DB21, ... DBX31.4 (REPOS mode change) is acknowledged with the existing interface signal, if the requested REPOS mode: DB21, ... DBX31.0-2 (REPOS mode) and the delay signal: DB31, ... DBX10.0 (REPOSDELAY) are taken over in the NC. The signal states refer to the current main run block.</td>
</tr>
</tbody>
</table>

Edge change 1 → 0
SERUPRO-ASUB stops automatically before REPOS and DB21, ... DBX31.4 (REPOS mode change) does not affect the SERUPRO approach.
Corresponding to ... DB21, ... DBX31.4 (REPOS mode-change)

DB21, ... DBX319.1 - DBX319.3
Active REPOS mode (A, B, C)

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meaning</td>
<td>The active REPOS mode is displayed with the interface signals A, B and C.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Active REPOS mode</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 = no REPOS mode active</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 = repositioning to block start point RMBBL</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2 = repositioning to interruption point RMIBL</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3 = repositioning to block end point RMEBL</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4 = repositioning to nearest path point RMNBL</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Active REPOS mode (A, B, C)

Example of the sequence of REPOS acknowledgements in the part program and signal timing of the acknowledgement process from the NC:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start the part program</td>
</tr>
<tr>
<td>2</td>
<td>Stop part program</td>
</tr>
<tr>
<td>3</td>
<td>Preselect RMNBL</td>
</tr>
<tr>
<td>4</td>
<td>Initiate ASUB</td>
</tr>
<tr>
<td>5</td>
<td>Command with ASUB is started</td>
</tr>
<tr>
<td>6</td>
<td>Repositioning motion from REPOS has been completed. The remaining block starts.</td>
</tr>
<tr>
<td>7</td>
<td>The remaining block is completed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corresponding to</th>
<th>DB21, ... DBX319.1 - DBX319.3 (REPOS mode)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DB21, ... DBX319.4 (REPOS mode-change)</td>
</tr>
<tr>
<td></td>
<td>DB21, ... DBX319.0 (acknowledgement of REPOS mode change)</td>
</tr>
<tr>
<td></td>
<td>DB31, ... DBX70.2 (acknowledgement of REPOS delay)</td>
</tr>
</tbody>
</table>

**Additional references**

See Section "Block search Type 5 (SERUPRO) (Page 493)"
Z1: NC/PLC interface signals

19.6 Mode group, channel, program operation, reset response (K1)

---

**DB21, ... DBX319.5**

**Repos DEFERAL Chan**

- **Edge evaluation:** No
- **Signal(s) updated:** Cyclically
- **Signal state 1:** All axes currently controlled by this channel have either no REPOS offset or their REPOS offsets are invalid.
- **Signal state 0:** Miscellaneous.
- **Corresponding to ...** DB31, ... DBX70.0 (Repos offset)

---

**DB21, ... DBB376**

**ProgEventDisplay**

- **Edge evaluation:** No
- **Signal(s) updated:** Event-driven
- **Note:** Signal duration of at least one complete PLC cycle
- **Meaning of individual bits:**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Part program start from channel state &quot;Reset&quot;</td>
</tr>
<tr>
<td>1</td>
<td>Part program end</td>
</tr>
<tr>
<td>2</td>
<td>Operator panel reset</td>
</tr>
<tr>
<td>3</td>
<td>Startup</td>
</tr>
<tr>
<td>4</td>
<td>1st start after block search</td>
</tr>
<tr>
<td>5 - 7</td>
<td>Reserved, currently always 0</td>
</tr>
</tbody>
</table>

- **Signal state 1:** The event assigned to the bit has activated the "Event-driven program call" function.
- **Signal state 0:** The event assigned to the bit has not activated the "Event-driven program call" function or the event-driven user program has expired or was canceled with RESET.

---

**DB21, ... DBX378.0**

**ASUB is active**

- **Edge evaluation:** No
- **Signal(s) updated:** Event-driven
- **Signal state 1:** One ASUB is active.
  - **Note:** The user gets a feedback on a running ASUB through DB21, … DBX378.0 even outside FC9 block.
- **Signal state 0:** No ASUB is active.

---

**DB21, ... DBX378.1**

**Still ASUB is active**

- **Edge evaluation:** No
- **Signal(s) updated:** Event-driven
- **Signal state 1:** An ASUB with suppressed display updating (refer to MD20191 $MC_IGN_PROG_STATE_ASUP) is active.
- **Signal state 0:** No ASUB with suppressed display updating is active.
### 19.6.5 Signals to axis/spindle (DB31, …)

<table>
<thead>
<tr>
<th>DB31, …</th>
<th>DBX10.0</th>
<th>REPOSDELAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: no</td>
<td>Signal(s) updated: Cyclically</td>
<td></td>
</tr>
<tr>
<td>Signal state 1</td>
<td>The REPOS offset of the axis is first applied with its next programming.</td>
<td></td>
</tr>
<tr>
<td>Signal state 0</td>
<td>There is no REPOS offset.</td>
<td></td>
</tr>
<tr>
<td>Special cases, errors, …..</td>
<td>The signal is not relevant for path axes.</td>
<td></td>
</tr>
</tbody>
</table>
| Corresponding to … | DB21, … DBX31.0 - DBX31.2 (REPOS mode)  
| | DB31, … DBX70.2 (REPOSDELAY acknowledge)  
| | DB31, … DBX72.0 (REPOSDELAY) |

### 19.6.6 Signals from axis/spindle (DB31, …)

<table>
<thead>
<tr>
<th>DB31, …</th>
<th>DBX70.0</th>
<th>REPOS offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: no</td>
<td>Signal(s) updated: cyclic</td>
<td></td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>A REPOS offset must be applied for the appropriate axis.</td>
<td></td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>No REPOS offset need be applied for the appropriate axis.</td>
<td></td>
</tr>
<tr>
<td>Corresponds to ….</td>
<td>DB31, … DBX70.1 (REPOS offset valid)</td>
<td></td>
</tr>
</tbody>
</table>

### DB31, … DBX70.1

<table>
<thead>
<tr>
<th>REPOS offset valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: no</td>
</tr>
</tbody>
</table>
| Signal state 1 or edge change 0 → 1 | The range of validity of the REPOS offset is indicated with the value 1.  
The REPOS offset was calculated to be valid. |
### DB31, ... DBX70.1 REPOS offset valid

<table>
<thead>
<tr>
<th>Signal state 0 or edge change&lt;br&gt;1 → 0</th>
<th>A value of zero indicates that the REPOS offset was calculated to be invalid.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application example(s)</td>
<td>Updating the REPOS offset in the range of validity:&lt;br&gt;Between SERUPRO end and start, the axis can be moved in JOG mode with a mode change. The user moves the REPOS offset to the zero value.</td>
</tr>
<tr>
<td>Corresponds to ...</td>
<td>DB31, ... DBX70.0 (REPOS offset)</td>
</tr>
</tbody>
</table>

### DB31, ... DBX70.2 REPOS Delay Ackn

<table>
<thead>
<tr>
<th>Edge evaluation: no&lt;br&gt;Signal(s) updated: cyclic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change&lt;br&gt;0 → 1&lt;br&gt;The axis was programmed within a traversing block, and the REPOS offset was applied. &lt;br&gt;Note&lt;br&gt;A REPOS offset was available for the axis, and REPOSDELAY was active:&lt;br&gt;DB31, ... DBX10.0 (REPOSDELAY) == 1 &lt;br&gt;This signal behaves the same as:&lt;br&gt;DB21, ... DBX319.1 - DBX319.3 (Repos Path Mode Ackn)</td>
</tr>
<tr>
<td>Signal state 0 or edge change&lt;br&gt;1 → 0&lt;br&gt;The value zero is used to acknowledge that the REPOS offset is not active for this axis. This signal is cancelled on activation of the remaining block.</td>
</tr>
<tr>
<td>Corresponds to ...</td>
</tr>
</tbody>
</table>

### DB31, ... DBX72.0 REPOSDELAY

<table>
<thead>
<tr>
<th>Edge evaluation: no&lt;br&gt;Signal(s) updated: cyclic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change&lt;br&gt;0 → 1&lt;br&gt;After a block search, a REPOS offset is applied for this axis. However it is not applied using the approach block, but rather using the next traversing block in which the axis is programmed.</td>
</tr>
<tr>
<td>Signal state 0 or edge change&lt;br&gt;1 → 0&lt;br&gt;The REPOS offset for this axis is not active.</td>
</tr>
<tr>
<td>Special cases, errors, ......&lt;br&gt;The signal is not relevant for path axes.</td>
</tr>
<tr>
<td>Corresponds to ...</td>
</tr>
</tbody>
</table>
19.7 Axis types, coordinate systems, frames (K2)

19.7.1 Signals to axis/spindle (DB31, ...)

<table>
<thead>
<tr>
<th>DB31, ...</th>
<th>Path axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: no</td>
<td>Signal(s) updated: cyclic</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>The axis is involved in the path (path axis).</td>
</tr>
<tr>
<td>Note</td>
<td>In conjunction with SERUPRO in status &quot;Target block found&quot;, the signal refers to the status of the axis in the target block.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>The axis is not involved in the path.</td>
</tr>
</tbody>
</table>

19.8 Emergency stop (N2)

19.8.1 Signals to NC (DB10)

<table>
<thead>
<tr>
<th>DB10</th>
<th>Emergency stop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>The NC is brought into the emergency stop state and the emergency stop sequence in the NC is started.</td>
</tr>
</tbody>
</table>
Emergency stop

**Signal state 0 or edge change 1 → 0**
The NC is not in the emergency stop state.
The emergency stop state is (still) active but can be reset using the interface signals:
DB10 DBX56.2 (acknowledge emergency stop)
and
DB11 DBX0.7 (mode group reset)

**Corresponding to**
DB10 DBX56.2 (acknowledge emergency stop)
DB10 DBX106.1 (emergency stop active)

---

**Acknowledge emergency stop**

**Signal state 1 or edge change 0 → 1**
The emergency stop state is only reset if the interface signal:
DB10 DBX56.2 (acknowledge emergency stop)
is set followed by the interface signal:
DB11, ... DBX0.7 (mode group reset).

It must be noted that IS "Acknowledge emergency stop" and IS "Reset" must be set (together) for a long enough period so that the interface signal:
DB10 DBX106.1 (emergency stop active)
was reset.

Resetting the emergency stop state has the following effects:
- The controller enable is switched in.
- Follow-up mode is canceled for all axes and position control mode resumed.
- DB31, ... DBX61.5 set (position controller active).
- DB11, ... DBX6.3 set (mode group ready).
- DB10 DBX106.1 reset (emergency stop active).
- Alarm 3000 is cleared.
- Part program processing is aborted for all channels.

---

**Diagram**

1. The IS "Acknowledge emergency stop" has no effect
2. The IS "Reset" has no effect
3. The IS "Acknowledge emergency stop" and "Reset" reset "Emergency stop active"
<table>
<thead>
<tr>
<th><strong>DB10</strong></th>
<th><strong>DBX56.2</strong></th>
<th><strong>Acknowledge emergency stop</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Special cases, errors, ......</td>
<td>The emergency stop state cannot be reset using the interface signal: DB21, ... DBX7.7 (reset).</td>
<td></td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB10 DBX56.1 (Emergency stop)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB10 DBX106.1 (emergency stop active)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB11 DBX0.7 (mode group reset)</td>
<td></td>
</tr>
</tbody>
</table>

### 19.8.2 Signals from NC (DB10)

<table>
<thead>
<tr>
<th><strong>DB10</strong></th>
<th><strong>DBX106.1</strong></th>
<th><strong>Emergency stop active</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
<td></td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>The NC is in the emergency stop state.</td>
<td></td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB10 DBX56.1 (Emergency stop)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB10 DBX56.2 (acknowledge emergency stop)</td>
<td></td>
</tr>
</tbody>
</table>

### 19.9 PLC basic program (P3)

To describe the NC/PLC interface signals, refer to:

**References:**
Functions Manual, Basic Functions; PLC Basic Program (P3)
Chapter: "Signal/Data Specifications"
## 19.10 Reference point approach (R1)

### 19.10.1 Signals to channel (DB21, ...)

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>DBX1.0</th>
<th><strong>Activate referencing</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td></td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>

**Signal state 1 or edge change 0 → 1**

Channel-specific referencing is started using the interface signal: DB21, ... DBX1.0 (activate referencing).

The control acknowledges a successful start using the interface signal: DB21, ... DBX33.0 (referencing active)

With the channel-specific referencing each machine axis, which is allocated to a channel, can be referenced (control internals are simulated by traversing keys plus/minus).

Using the axis-specific machine data:

- MD34110 $MA_REFP_CYCLE_NR (axis sequence for channel-specific referencing)

  * can determine in which sequence the machine axes are referenced.

  * When all of the axes entered in REFP_CYCLE_NO have reached their reference point, the interface signal:

    DB21, ... DBX36.3 (all axes stationary)

  is set.

**Application exam‐ple(s)**

If the machine axes are to be referenced in a particular sequence, there are the following possibilities:

- The operator must observe the correct sequence when starting.
- The PLC must check the sequence when starting or define it itself.
- The channel-specific referencing function will be used.

**Corresponding to ...**

- DB21, ... DBX33.0 (activate referencing)
- DB21, ... DBX36.2 (all axes with obligatory reference point are referenced)

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>DBX33.0</th>
<th><strong>Referencing active</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td></td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>

**Signal state 1 or edge change 0 → 1**

The channel-specific referencing was started using the interface signal: DB21, ... DBX1.0 (activate referencing) and the successful start was acknowledged using the interface signal: DB21, ... DBX33.0 (referencing active).

The channel-specific referencing is operational.

**Signal state 0 or edge change 1 → 0**

- Channel-specific referencing is completed
- Axis-specific referencing is running
- No referencing active

**Signal irrelevant for ...**

Spindles

**Corresponding to ...**

DB21, ... DBX1.0 (activate referencing)
### 19.10.2 Signals from channel (DB21, ...)

| DB21, ...  |
| DBX36.2    | All axes that have to be referenced are referenced |

<table>
<thead>
<tr>
<th>Edge evaluation: no</th>
<th>Signal(s) updated: cyclic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>All axes that must be referenced (linear axes and rotary axes) of the channel are referenced. The machine data: MD20700 $MC_REFP_NC_START_LOCK (NC start inhibit without reference point) is zero. If two position measuring systems are connected to an axis, that would prevent an NC start, then the active one must be referenced so that the axis is considered to have been referenced. An NC Start command for parts program processing is only accepted when this signal is present. Axes that have to be referenced are axes, if: MD34110 $MA_REFP_CYCLE_NR = -1 and the axis is not in the parked position (position measuring system inactive and the controller enable withdrawn).</td>
</tr>
</tbody>
</table>

| Signal state 0 or edge change 1 → 0 | One or more axes of the channel have not been referenced. |

| Special cases, errors, ....... | The spindles of the channel have no effect on this interface signal. |

| Corresponding to ... | DB31, ... DBX60.4 (referenced / synchronized 1) DB31, ... DBX60.5 (referenced / synchronized 2) |

### 19.10.3 Signals to axis/spindle (DB31, ...)

| DB31, ...  |
| DBX2.4 - DBX2.7 | Reference point value 1 to 4 |

<table>
<thead>
<tr>
<th>Edge evaluation: no</th>
<th>Signal(s) updated: cyclic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>When the reference cam is reached, the NCK is signaled which coded reference cam is actuated. The interface signal: DB31, ... DBX2.4 - DBX2.7 (reference point values 1 to 4) must remain set until the reference point is reached, or until a new coded reference cam is actuated. If the machine axis has reached the reference point (axis stationary) then using the reference point value pre-selected using IS &quot;reference point values 1 to 4&quot; from the machine data: MD34100 $MA_REFP_SET_POS (reference point value) is transferred into the control as the new reference position.</td>
</tr>
</tbody>
</table>

| Signal state 0 or edge change 1 → 0 | No effect. |

| Signal irrelevant for ... | Linear measurement systems with distance-coded reference marks |

| Application example(s) | On a machine tool with large traversing distances, four coded reference cams can be distributed over the entire distance traveled by the axis, four different reference points approached and the time required to reach a valid referenced point reduced. |
### Reference point value 1 to 4

<table>
<thead>
<tr>
<th>DB31, ...</th>
<th>Reference point value 1 to 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX2.4 - DBX2.7</td>
<td>If the machine axis has arrived at the reference point and none of the four &quot;reference point value 1 to 4&quot; interface signals has been set, the value of the reference point is automatically set to 1.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>MD34100 $MA_REFP_SET_POS (reference point value)</td>
</tr>
</tbody>
</table>

### Reference point approach delay

<table>
<thead>
<tr>
<th>DB31, ...</th>
<th>Reference point approach delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX12.7</td>
<td>Edge evaluation: no</td>
</tr>
<tr>
<td></td>
<td>Signal(s) updated: cyclic</td>
</tr>
<tr>
<td></td>
<td>Signal state 1 or edge change 0 → 1</td>
</tr>
<tr>
<td></td>
<td>Signal state 0 or edge change 1 → 0</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB31, ... DBX2.4 - DBX2.7 (reference point values 1 to 4)</td>
</tr>
</tbody>
</table>

### Signals from axis/spindle (DB31, ...)

<table>
<thead>
<tr>
<th>DB31, ...</th>
<th>Referenced/synchronized 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBX80.4</td>
<td>Edge evaluation:</td>
</tr>
<tr>
<td></td>
<td>Signal(s) updated:</td>
</tr>
<tr>
<td></td>
<td>Axes:</td>
</tr>
<tr>
<td></td>
<td>When being referenced, if the machine axis has reached the reference point (incremental measuring systems) or the target point (for length measuring system with distance-coded reference marks), then the machine axis is referenced and the following interface signal is set: DB31, ... DBX80.4 (referenced/synchronized 1) (depending on which position measuring system is active when referencing).</td>
</tr>
<tr>
<td></td>
<td>Spindles:</td>
</tr>
<tr>
<td></td>
<td>After power-on, a spindle is synchronized the latest after one spindle revolution (360 degrees) (the zero mark passed or the Bero responded).</td>
</tr>
<tr>
<td></td>
<td>Signal state 1 or edge change 0 → 1</td>
</tr>
<tr>
<td></td>
<td>The machine axis/spindle with position measuring system 1 is not referenced/synchronized.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB31, ... DBX1.5 (position measuring system 1)</td>
</tr>
</tbody>
</table>
### Referenced/synchronized 2

<table>
<thead>
<tr>
<th>DB31, ... DBX60.5</th>
<th>Referenced/synchronized 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation:</td>
<td>Signal(s) updated:</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>Axes: When being referenced, if the machine axis has reached the reference point (incremental measuring systems) or the target point (for length measuring systems with distance-coded reference marks), then the machine axis is referenced and the following interface signal is set: DB31, ... DBX60.5 (referenced/synchronized 2) (depending on which position measuring system is active when referencing).</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>Spindles: After power-on, a spindle is synchronized the latest after one spindle revolution (360 degrees) (the zero mark passed or the Bero responded). The machine axis/spindle with position measuring system 2 is not referenced/synchronized.</td>
</tr>
</tbody>
</table>

### Corresponding to ... DB31, ... DBX1.6 (position measuring system 2)

MD34102 $MA_REFP_SYNC_ENCS (measuring system calibration) = 0

### POS_RESTORED 1

<table>
<thead>
<tr>
<th>DB31, ... DBX71.4</th>
<th>POS_RESTORED 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation:</td>
<td>Signal(s) updated:</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>If MD34210 $MA_ENC_REFP_STATE is set to a value of 3, the last axis position buffered before switch off is restored in distance-coded, incremental measuring systems. Referencing does not take place automatically. Position measuring system 1 is in &quot;Position restored&quot; state.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>The machine axis/spindle with position measuring system 1 is not restored.</td>
</tr>
</tbody>
</table>

### Corresponding to ... DB31, ... DBX60.4 (referenced / synchronized 1)

### POS_RESTORED 2

<table>
<thead>
<tr>
<th>DB31, ... DBX71.5</th>
<th>POS_RESTORED 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation:</td>
<td>Signal(s) updated:</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>If MD34210 $MA_ENC_REFP_STATE is set to a value of 3, the last axis position buffered before switch off is restored in distance-coded, incremental measuring systems. Referencing does not take place automatically. Position measuring system 2 is in &quot;Position restored&quot; state.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>The machine axis/spindle with position measuring system 2 is not restored.</td>
</tr>
</tbody>
</table>

### Corresponding to ... DB31, ... DBX60.5 (referenced / synchronized 2)
19.11 Spindles (S1)

19.11.1 Signals to axis/spindle (DB31, ...)

<table>
<thead>
<tr>
<th>DB31, ... DBX2.2</th>
<th>Spindle reset/delete distance to go</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>

**Edge change 0 → 1**

- Independent of the machine data: MD35040 $MA_SPIND_ACTIVE_AFTER_RESET selects a spindle reset for the various spindle operating modes in the following fashion:
  - **Control mode:**
    - Spindle stops
    - Program continues to run
    - Spindle continues to run with subsequent M and S program commands
  - **Oscillating mode:**
    - Oscillation is interrupted
    - Axes continue to run
    - Program continues with the actual gearbox stage
    - With the following M value and higher S value, it is possible that the IS: DB31, ... DBX83.1 (programmed speed high) is set.
  - **Positioning mode:**
    - Is stopped
  - **Axis operation:**
    - Is stopped

**Edge change 1 → 0**

- No effect.

**Corresponding to ...**

MD35040 $MA_SPIND_ACTIVE_AFTER_RESET (own spindle reset)
DB21, ... DBX7.7 (reset)
DB31, ... DBX2.2 (delete distance to go): is a different name for the same signal
If the new gear stage is engaged, the PLC user program sets the interface signals:
DB31, ... DBX16.2 - DBX16.0 (actual gear stage A to C)
and
DB31, ... DBX16.3 (gear is changed over).
This signals the NCK that the correct gear stage has been successfully engaged.
The gear change is considered to have been completed (spindle oscillation mode is deselected), the
spindle accelerates in the new gear stage to the last programmed spindle speed and the next block
in the parts program can be executed.
The actual gear stage is output in coded format.
For each of the 5 gear stages, there is one set of parameters assigned as follows:

<table>
<thead>
<tr>
<th>Parameter set No.</th>
<th>NC/PLC interface</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>–</td>
<td>Data for axis mode</td>
</tr>
<tr>
<td>1</td>
<td>000</td>
<td>Data for 1st gear stage</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>Data for 2nd gear stage</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>Data for 3rd gear stage</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>Data for 4th gear stage</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>Data for 5th gear stage</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td></td>
</tr>
<tr>
<td></td>
<td>111</td>
<td></td>
</tr>
</tbody>
</table>

If the PLC user reports back to the NCK with a different actual gear stage than issued by the NCK as
the setpoint gear stage, the gear change is still considered to have been successfully completed and
the actual gear stage A to C is activated.

DB31, ... DBX82.0 - DBX82.2 (setpoint gear stage A to C)
DB31, ... DBX82.3 (change over gear stage)
DB31, ... DBX16.3 (gear is changed)
DB31, ... DBX18.5 (oscillation speed)
Parameter sets for gear stages
### Z1: NC/PLC interface signals

#### 19.11 Spindles (S1)

<table>
<thead>
<tr>
<th>DB31, ... DBX16.3</th>
<th>Gear is changed over</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>When the new gear stage is engaged, the PLC user sets the interface signals: DB31, ... DBX16.0 - DBX16.2 (actual gear stages A to C) and DB31, ... DBX16.3 (gear is changed over). This signals the NCK that the correct gear stage has been successfully engaged. The gear stage change is complete (spindle oscillation mode is deselected), the spindle accelerates in the new gear stage to the last programmed spindle speed and the next block in the parts program can be executed. The interface signal: DB31, ... DBX82.3 (change over gear) is reset by the NCK - the PLC user then resets the interface signal: (gear is changed over).</td>
</tr>
<tr>
<td>Edge change 1 → 0</td>
<td>No effect.</td>
</tr>
<tr>
<td>Signal irrelevant for ...</td>
<td>... spindle modes other than the oscillation mode</td>
</tr>
<tr>
<td>Special cases, errors, ......</td>
<td>If the PLC user reports back to the NCK with a different actual gear stage than issued by the NCK as the setpoint gear stage, the gear change is still considered to have been successfully completed and the actual gear stage A to C is activated.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB31, ... DBX16.2 - DBX16.0 (actual gear stage A to C) DB31, ... DBX82.2 - DBX82.0 (setpoint gear stage A to C) DB31, ... DBX82.3 (change over gear stage) DB31, ... DBX18.5 (oscillation speed)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB31, ... DBX16.4 - DBX16.5</th>
<th>Resynchronizing spindles 1 and 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>The spindle should be resynchronized, as the synchronization between the position measuring system of the spindle and the 0 degree position has been lost.</td>
</tr>
<tr>
<td>Edge change 1 → 0</td>
<td>No effect.</td>
</tr>
<tr>
<td>Signal irrelevant for ...</td>
<td>... spindle modes other than the control mode.</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>The machine has a selector switch to changeover between a vertical and a horizontal spindle. Two different position encoders are used (one for the vertical spindle and one for the horizontal spindle), but only one actual value input is used on the control. When the system switches from the vertical to the horizontal spindle, the spindle must be resynchronized. This synchronization is triggered by the IS &quot;re-synchronize spindle 1 or 2&quot;.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB31, ... DBX60.4 (referenced / synchronized 1) DB31, ... DBX60.5 (referenced / synchronized 2)</td>
</tr>
</tbody>
</table>
### 19.11 Spindles (S1)

<table>
<thead>
<tr>
<th>DB31, ... DBX16.7</th>
<th>Delete S value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>Control mode:</td>
</tr>
<tr>
<td>Control mode:</td>
<td>• Spindle stops</td>
</tr>
<tr>
<td>Control mode:</td>
<td>• Program continues to run</td>
</tr>
<tr>
<td>Oscillation mode, axis mode, positioning mode:</td>
<td>• Spindle continues to run with the following S value, if M3 or M4 were active</td>
</tr>
<tr>
<td>Edge change 1 → 0</td>
<td>No effect.</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Terminating traversing motion on account of an external signal (e.g. sensing probe).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB31, ... DBX17.4 - DBX17.5</th>
<th>Re-synchronizing spindle when positioning 1 and 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>When positioning, the spindle must be re-synchronized.</td>
</tr>
<tr>
<td>Edge change 1 → 0</td>
<td>No effect.</td>
</tr>
<tr>
<td>Signal irrelevant for ...</td>
<td>... spindle modes other than the positioning mode.</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>The spindle has an indirect measuring system and slippage may occur between the motor and the clamp. If the signal=1 - when positioning is started, the old reference is deleted and the zero mark is searched for again before the end position is approached.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB31, ... DBX60.4 (referenced / synchronized 1)</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB31, ... DBX60.5 (referenced / synchronized 2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB31, ... DBX17.6</th>
<th>Invert M3/M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>The direction of rotation of the spindle motor changes for the following functions:</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>• M3</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>• M4</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>• M5</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>• SPOS/M19/SPOSA from the motion; not effective for SPOS/M19/SPOSA from zero speed (stationary).</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>The machine has a selector switch to changeover between a vertical and a horizontal spindle. The mechanical design is implemented so that for the horizontal spindle, one more gearwheel is engaged than for the vertical spindle. The direction of rotation must therefore be changed for the vertical spindle if the spindle is always to rotate clockwise with M3.</td>
</tr>
</tbody>
</table>
If the interface signal: DB31, ... DBX18.4 (oscillation controlled by the PLC) is not set, then automatic oscillation in the NCK is carried out using the interface signal: DB31, ... DBX18.5 (oscillation speed).

The two times for the directions of rotation are entered in the machine data:
- MD35440 $MA_SPIND_OSCILL_TIME_CW (oscillation time for the M3 direction)
- MD35450 $MA_SPIND_OSCILL_TIME_CCW (oscillation time for the M4 direction).

If the IS "oscillation via PLC" is set, then with the IS "oscillation speed" a speed is only output in conjunction with the interface signals:
- DB31, ... DBX18.6 - DBX18.7 (setpoint direction of rotation, counter-clockwise and clockwise).

The oscillation, i.e., the continuous change of the direction of rotation, is performed by the PLC user program using the interface signal "setpoint direction of rotation, counter-clockwise and clockwise" (oscillation via the PLC).

If the new gear stage cannot be engaged in spite of several attempts by the NCK, the system can be switched to oscillation via PLC. Both of the times for the directions of rotation can then be altered by the PLC user program as required. This ensures that the gear stage is reliably changed - even with unfavorable gear wheel positions.

Corresponding to ...
- MD35440 $MA_SPIND_OSCILL_TIME_CW (oscillation time for the M3 direction)
- MD35450 $MA_SPIND_OSCILL_TIME_CCW (oscillation time for the M4 direction)
- DB31, ... DBX18.5 (oscillation speed)
- DB31, ... DBX18.6 (setpoint direction of rotation, counter-clockwise)
- DB31, ... DBX18.7 (setpoint direction of rotation, clockwise)
### Oscillation enable

<table>
<thead>
<tr>
<th>DB31, ... DBX18.5</th>
<th>Oscillation enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>

**Signal state 1**

If the gear stage is to be changed (DB31, ... DBX82.3 (change over gear) is set), then the spindle operating mode changes to the oscillation mode.

Depending on the instant in time that the interface signal: DB31, ... DBX18.5 (oscillation speed) is set, the spindle decelerates down to standstill with different deceleration levels:

The IS "Oscillation speed" is set before the interface signal: DB31, ... DBX82.3 (change over gear) is set by the NCK.

When oscillating, the spindle is decelerated down to standstill with the deceleration: MD35410 $MA_SPIND_OSCILL_ACCEL.

Once the spindle is stationary, oscillation is immediately initiated.

The IS "Oscillation speed" is enabled after the IS "Change gear" is set by the NCK and when the spindle is stationary. The position controller is disabled. The spindle decelerates with the specified deceleration rate in the speed controlled mode.

After the IS "oscillation speed" is set, the spindle starts to oscillate with the oscillation acceleration (MD35410).

if the interface signal: DB31, ... DBX18.4 (oscillation via the PLC) is not set, then automatic oscillation is executed in the NCK using the IS "Oscillation speed".

The two times for the directions of rotation are entered in the machine data:

MD35440 $MA_SPIND_OSCILL_TIME_CW (oscillation time for the M3 direction)

and

MD35450 $MA_SPIND_OSCILL_TIME_CCW (oscillation time for the M4 direction).

If the IS "oscillation via PLC" is set, then with the IS "oscillation speed" a speed is only output in conjunction with the interface signals:

DB31, ... DBX18.6 - DBX18.7 (setpoint direction of rotation, counter-clockwise and clockwise).

The oscillation, i.e. the continuous change of the direction of rotation, is performed by the PLC user program using the interface signal "setpoint direction of rotation, counter-clockwise and clockwise" (oscillation via the PLC).

**Signal state 0**

The spindle does not oscillate.

**Signal irrelevant for ...**

... All spindle modes except the oscillation mode.

**Application example(s)**

The oscillation speed is used to make it easier to engage a new gear stage.

**Corresponding to ...**

DB31, ... DBX18.4 (oscillation controlled by the PLC)

DB31, ... DBX18.7 (setpoint direction of rotation, counter-clockwise)

DB31, ... DBX18.6 (setpoint direction of rotation, clockwise)
### 19.11 Spindles (S1)

#### DB31, ... DBX18.6 - DBX18.7

<table>
<thead>
<tr>
<th>Edge evaluation: Yes</th>
<th>Oscillation direction of rotation counter-clockwise / oscillation direction of rotation clockwise</th>
</tr>
</thead>
</table>
| Edge change 0 → 1   | If the interface signal:

- DB31, ... DBX18.4 (oscillation by the PLC) is set, then the direction of rotation for the oscillation speed can be entered using the two interface signals:

- DB31, ... DBX18.6 - DBX18.7 (setpoint direction of rotation counter-clockwise and clockwise).

The times for the oscillation movement of the spindle motor are defined by setting the interface signals "direction of rotation setpoint counter-clockwise and clockwise" for a corresponding length of time. |
| Signal irrelevant for ... | ... spindle modes other than the oscillation mode |
| Application example(s) | Refer to DB31, ... DBX18.4 (oscillation controlled by the PLC) |
| Special cases, errors, ...... | If both of the interface signals are set simultaneously enabled, no oscillation speed is output. If an interface signal is not set, then an oscillation speed is not output. |
| Corresponding to ... | DB31, ... DBX18.4 (oscillation controlled by the PLC) DB31, ... DBX18.5 (oscillation speed) |

#### 19.11.2 Signals from axis/spindle (DB31, ...)

<table>
<thead>
<tr>
<th>DB31, ... DBX60.0</th>
<th>Spindle/no axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
</tbody>
</table>
| Edge change 0 → 1 | The machine axis is operated in one of the following spindle modes:

- Open-loop control mode
- Oscillation mode
- Positioning mode
- Tapping without compensating chuck
- Synchronous mode

The interface signals to the axis (DB31, ... DBB12 - DBB15) and from the axis (DB31, ... DBB74 - DBB81) are invalid.

The interface signals to the spindle (DB31, ... DBB16 - DBB19) and from the spindle (DB31, ... DBB82 - DBB91) are invalid. |
| Edge change 1 → 0 | The machine axis is operated as rotary axis.

The interface signals to the axis (DB31, ... DBB12 - DBB15) and from the axis (DB31, ... DBB74 - DBB81) are valid.

The interface signals to the spindle (DB31, ... DBB16 - DBB19) and from the spindle (DB31, ... DBB82 - DBB91) are invalid. |
| Application example(s) | If a machine axis is operated alternating as spindle or rotary axis, using the signal it is possible to identify which mode is currently active. Application examples:

- Turning machine: Spindle ↔ C axis
- Milling machine: Spindle ↔ rotary axis for rigid tapping |
### DB31, ... DBX82.0 - 82.2

<table>
<thead>
<tr>
<th>Setpoint gear stage A to C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Edge change 0 → 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>The set gear stage is output in coded format:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2 (C)</th>
<th>Bit 1 (B)</th>
<th>Bit 0 (A)</th>
<th>Gear stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>invalid value</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>invalid value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal irrelevant for ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>... Other spindle modes except oscillation mode</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corresponding to ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB31, ... DBX82.3 (change over gear stage)</td>
</tr>
<tr>
<td>DB31, ... DBX16.0 - DBX16.2 (actual gear stage A to C)</td>
</tr>
<tr>
<td>DB31, ... DBX16.3 (gear is changed)</td>
</tr>
</tbody>
</table>

### DB31, ... DBX82.3

<table>
<thead>
<tr>
<th>Change gear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specification of gear stage:</th>
</tr>
</thead>
</table>

- Manual specification using M function M41 - M45 corresponding to gear stage 1 - 5
  - If set gear stage <> actual gear stage =>
    - DB31, ... DBX82.3 (change gear) = 1
    - DB31, ... DBX82.0 - DBX82.2 (set gear stage) = set gear stage
- Automatic gear stage selection depending on the progr. Spindle speed via M-function M40
  - at specified setpoint speed requires gear stage change =>
    - DB31, ... DBX82.3 (change gear) = 1
    - DB31, ... DBX82.0 - DBX82.2 (setpoint gear stage) = setpoint gear stage

<table>
<thead>
<tr>
<th>Edge change 0 → 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>New set gear stage was specified AND set gear stage &lt;&gt; actual gear stage</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Special cases, errors, .......</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal is not output if: Set gear stage == actual gear stage</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corresponding to ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB31, ... DBX82.0 - DBX82.2 (setpoint gear stage A to C)</td>
</tr>
<tr>
<td>DB31, ... DBX16.0 - DBX16.2 (actual gear stage A to C)</td>
</tr>
<tr>
<td>DB31, ... DBX16.3 (gear is changed)</td>
</tr>
</tbody>
</table>
### DB31, DBX83.0

<table>
<thead>
<tr>
<th><strong>Speed limit exceeded</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
</tr>
<tr>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td><strong>Edge change 0 → 1</strong></td>
</tr>
<tr>
<td>The actual speed exceeds the maximum spindle speed:</td>
</tr>
<tr>
<td>MD35100 $MA_SPIND_VELO_LIMIT</td>
</tr>
<tr>
<td>by more than the spindle speed tolerance:</td>
</tr>
<tr>
<td>MD35150 $MA_SPIND_DES_VELO_TOL</td>
</tr>
</tbody>
</table>

**Corresponding to**
- MD35150 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance)
- MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed)

### DB31, DBX83.1

<table>
<thead>
<tr>
<th><strong>Setpoint speed limited (programmed speed too high)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
</tr>
<tr>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td><strong>Edge change 0 → 1</strong></td>
</tr>
<tr>
<td>The effective setpoint speed exceeds the actual max. limit value. The setpoint speed is limited to this limit.</td>
</tr>
<tr>
<td>Limit values:</td>
</tr>
<tr>
<td>· MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT (maximum speed of gear stage)</td>
</tr>
<tr>
<td>· MD35100 $MA_SPIND_VELO_LIMIT (maximum spindle speed)</td>
</tr>
<tr>
<td>· DB31, DBX3.6 (spindle speed limitation to MD35160 $MA_SPIND_EXTERN_VELO_LIMIT)</td>
</tr>
<tr>
<td>· G26 (upper spindle speed limitation)</td>
</tr>
<tr>
<td>· LIMS (speed limitation for the master spindle if G96/G961/G97 is active)</td>
</tr>
<tr>
<td>· VELOLIM: Programmed spindle speed limitation in the open-loop speed controlled mode</td>
</tr>
<tr>
<td>· Safety Integrated</td>
</tr>
<tr>
<td>MD36931 $MA_SAFE_VELO_LIMIT (limit value for safely-reduced speed)</td>
</tr>
</tbody>
</table>

**Edge change 1 → 0**
The effective setpoint speed of the spindle is outside the maximum limit value.
### Setpoint speed limited (programmed speed too high)

<table>
<thead>
<tr>
<th>Application example(s)</th>
<th>In the PLC user program, it can be identified via the interface signal that the spindle setpoint speed has not been reached. Possible responses:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Indicate that the status is permissible and enable path feed: DB21, ... DBX6.0 = 0 (feed disable)</td>
</tr>
<tr>
<td></td>
<td>• Disable path feed or entire channel: DB21, ... DBX6.0 = 1 (feed disable)</td>
</tr>
<tr>
<td></td>
<td>DB31, ... DBX83.5 (spindle in setpoint range) is processed</td>
</tr>
</tbody>
</table>

### Safety Integrated

In addition to the limit value MD36931 $MA_SAFE_VELO_LIMIT, depending on the active safety speed level SG1 ... SGn, the following machine data should be taken into account:

- MD36932 $MA_SAFE_VELO_OVR_FACTOR
- MD36933 $MA_SAFE_DES_VELO_LIMIT

**Example:**

All standard limit values are greater than 1500 rpm.

- SG1 is active
- MD36932 $MA_SAFE_VELO_OVR_FACTOR[<SG1>] = 1111.11111 [rpm]
- MD36933 $MA_SAFE_DES_VELO_LIMIT[<SG1>] = 90%

**Programming:** M3 S1500

The speed setpoint is limited to 1000 rpm (MD36932 * MD36933).

**Corresponding to:**

- DB21, ... DBX6.0 (feed disable)
- DB31, ... DBX4.3 (feed/spindle stop)
- DB31, ... DBX83.5 (spindle in setpoint range)

### Setpoint speed increased (programmed speed too low)

<table>
<thead>
<tr>
<th>Edge evaluation: Yes</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge change 0 → 1</td>
<td>The effective setpoint speed is below the current min. limit value. The setpoint speed is limited to this limit. Limit values:</td>
</tr>
<tr>
<td></td>
<td>• MD35120 $MA_GEAR_STEP_MIN_VELO (minimum speed for automatic gear stage selection M40)</td>
</tr>
<tr>
<td></td>
<td>• MD35140 $MA_GEAR_STEP_MIN_VELO_LIMIT (minimum speed of the gear stage)</td>
</tr>
<tr>
<td></td>
<td>• G25 (lower spindle speed limitation)</td>
</tr>
</tbody>
</table>

**Edge change 1 → 0**

The set speed of the spindle is outside the minimum limit value.
### Z1: NC/PLC interface signals

#### 19.11 Spindles (S1)

<table>
<thead>
<tr>
<th>DB31, ... DBX83.2</th>
<th>Setpoint speed increased (programmed speed too low)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application example(s)</td>
<td>In the PLC user program, it can be identified via the interface signal that the spindle setpoint speed has not been reached. Possible responses:</td>
</tr>
<tr>
<td></td>
<td>• Indicate that the status is permissible and enable path feed: DB21, ... DBX6.0 = 0 (feed disable)</td>
</tr>
<tr>
<td></td>
<td>• Disable path feed or entire channel: DB21, ... DBX6.0 = 1 (feed disable)</td>
</tr>
<tr>
<td></td>
<td>DB31, ... DBX83.5 (spindle in setpoint range) is processed</td>
</tr>
<tr>
<td></td>
<td>The interface signal indicates if the programmed set speed is unattainable. The feed can be enabled nonetheless by means of the PLC user program. The PLC user can accept this state as permissible and enable the path velocity, or can disable the path velocity or the entire channel.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB21, ... DBX6.0 (feedrate disable)</td>
</tr>
<tr>
<td></td>
<td>DB31, ... DBX4.3 (feed/spindle stop)</td>
</tr>
<tr>
<td></td>
<td>DB31, ... DBX83.5 (spindle in setpoint range)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB31, ... DBX83.5</th>
<th>Spindle in setpoint range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>The actual speed of the spindle deviates from the set speed by less than the spindle speed tolerance: MD35150 $MA_SPIND_DES_VELO_TOL.</td>
</tr>
<tr>
<td>Edge change 1 → 0</td>
<td>The actual speed of the spindle deviates from the set speed by more than the spindle speed tolerance: MD35150 $MA_SPIND_DES_VELO_TOL.</td>
</tr>
<tr>
<td>Normal status during the acceleration/deceleration phase of the spindle.</td>
<td></td>
</tr>
<tr>
<td>Signal irrelevant for ...</td>
<td>... All spindle modes except for control mode (speed mode).</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Feed enable in the channel first with the end of the acceleration phase of the spindle: IF ( DB31, ... DBX83.5 (spindle in the setpoint range) == 1 ) THEN ( DB21, ... DBX6.0 (feed inhibit) = 0 ) ELSE ( DB21, ... DBX6.0 (feed inhibit) = 1 ) Note: With the feed disable, the positioning axes are also stopped.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>MD35500 $MA_SPIND_ON_SPEED_AT_IPO_START</td>
</tr>
<tr>
<td></td>
<td>MD35500 $MA_SPIND_DES_VELO_TOL (spindle speed tolerance)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB31, ... DBX83.7</th>
<th>Actual direction of rotation clockwise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Interface signal is only valid if the spindle is rotating: DB31, ... DBX61.4 (axis/spindle stationary) == 0 The actual direction of rotation is derived from the position measuring encoder.</td>
<td></td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>Actual direction of rotation: Clockwise</td>
</tr>
<tr>
<td>Edge change 1 → 0</td>
<td>Actual direction of rotation: Counterclockwise</td>
</tr>
</tbody>
</table>
### Actual direction of rotation clockwise

<table>
<thead>
<tr>
<th>DB31, ... DBX83.7</th>
<th>Signal irrelevant for ...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Spindle is stationary: DB31, ... DBX61.4 (axis/spindle stationary) == 1</td>
</tr>
<tr>
<td></td>
<td>- Spindles without position measuring encoder</td>
</tr>
<tr>
<td></td>
<td>Corresponding to ... DB31, ... DBX61.4 (axis/spindle stationary)</td>
</tr>
</tbody>
</table>

### Rigid tapping active

<table>
<thead>
<tr>
<th>DB31, ... DBX84.3</th>
<th>Rigid tapping active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>The spindle is internally switched to axis mode by the &quot;Rigid tapping&quot; function (G331/G332). This results in a reaction to or updating of the spindle-specific interface signals:</td>
<td></td>
</tr>
<tr>
<td>- DB31, ... DBX2.2 (spindle reset)</td>
<td></td>
</tr>
<tr>
<td>- DB31, ... DBX16.4 - DBX16.5 (synchronize spindle)</td>
<td></td>
</tr>
<tr>
<td>- DB31, ... DBX17.6 (invert M3/M4)</td>
<td></td>
</tr>
<tr>
<td>- DB31, ... DBX83.5 (spindle in setpoint range)</td>
<td></td>
</tr>
<tr>
<td>- DB31, ... DBX83.1 (programmable speed too high)</td>
<td></td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>Rigid tapping active.</td>
</tr>
<tr>
<td>Notice! If the following signals are set during rigid tapping, the thread will be destroyed:</td>
<td></td>
</tr>
<tr>
<td>- DB11, ... DBX0.7 (mode group reset) = 1</td>
<td></td>
</tr>
<tr>
<td>- DB21, ... DBX7.7 (channel reset) = 1</td>
<td></td>
</tr>
<tr>
<td>- DB31, ... DBX2.1 (controller enable) = 0</td>
<td></td>
</tr>
<tr>
<td>- DB31, ... DBX4.3 (feed stop) = 1</td>
<td></td>
</tr>
</tbody>
</table>

### Active spindle mode: Positioning mode

<table>
<thead>
<tr>
<th>DB31, ... DBX84.5</th>
<th>Active spindle mode: Positioning mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>Positioning mode (SPOS or SPOSA) is active.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB31, ... DBX84.7 (spindle operating mode, control mode)</td>
</tr>
<tr>
<td></td>
<td>DB31, ... DBX84.6 (spindle operating mode, oscillation mode)</td>
</tr>
</tbody>
</table>

### Active spindle mode: Oscillation mode

<table>
<thead>
<tr>
<th>DB31, ... DBX84.6</th>
<th>Active spindle mode: Oscillation mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>Oscillation mode is active.</td>
</tr>
<tr>
<td>Note: The spindle changes automatically to oscillation mode if there is a gear change.</td>
<td></td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB31, ... DBX84.7 (spindle operating mode, control mode)</td>
</tr>
<tr>
<td></td>
<td>DB31, ... DBX84.5 (spindle operating mode, positioning mode)</td>
</tr>
<tr>
<td></td>
<td>DB31, ... DBX82.3 (change over gear stage)</td>
</tr>
</tbody>
</table>
### 19.11 Spindles (S1)

<table>
<thead>
<tr>
<th>DB31, ... DBX84.7</th>
<th><strong>Active spindle mode: Open-loop control mode</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>The spindle is in control mode with the following functions:</td>
</tr>
<tr>
<td></td>
<td>• Spindle direction of rotation specification M3/M4 or spindle stop M5</td>
</tr>
<tr>
<td></td>
<td>• M41...M45, or automatic gear stage change M40</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB31, ... DBX84.6 (spindle operating mode, oscillation mode)</td>
</tr>
<tr>
<td></td>
<td>DB31, ... DBX84.5 (spindle operating mode, positioning mode)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB31, ... DBX85.5</th>
<th><strong>Spindle in position</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>The interface signal is processed exclusively with the function spindle positioning. This includes:</td>
</tr>
<tr>
<td></td>
<td>• SPOS, SPOSA and M19 in the part program</td>
</tr>
<tr>
<td></td>
<td>• SPOS and M19 in synchronized actions</td>
</tr>
<tr>
<td></td>
<td>• Spindle positioning, using FC18</td>
</tr>
<tr>
<td></td>
<td>• Spindle positioning via PLC interface (DB31, ... DBX30.4)</td>
</tr>
<tr>
<td>Requirement for the output of the DB31, ... DBX85.5 signal (spindle in position) is reaching the &quot;Exact stop fine&quot;.</td>
<td>DB31, ... DBX60.7 (exact stop fine) = 1</td>
</tr>
<tr>
<td>Additionally, the last programmed spindle position must have been reached on the setpoint side. If the spindle is already at the programmed position after a positioning, then signal DB31,... DBX85.5 (spindle in position) is set.</td>
<td></td>
</tr>
<tr>
<td>Edge change 1 → 0</td>
<td>When signal DB31, ... DBX60.7 is withdrawn (exact stop fine), then signal DB31, ... DBX85.5 (spindle in position) is also always reset.</td>
</tr>
<tr>
<td>Application example(s)</td>
<td>Spindle in position for the tool change</td>
</tr>
<tr>
<td></td>
<td>If the machine operator interrupts the tool change cycle (e.g. with NC stop, NC stop axis plus spindle, mode group stop, etc.), then NC/PLC interface signal DB31, ... DBX85.5 can be used to query that the position has been reached with which the spindle should enter the tool changer.</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB31, ... DBX60.7 (exact stop fine)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB31, ... DBB86</th>
<th><strong>M function for spindle</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: Yes</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td>Edge change 0 → 1</td>
<td>From the NCK one of the following M functions: M3, M4, M5, M19, M70 is output to the PLC. The output is performed via: See &quot;Corresponds to ...&quot; below</td>
</tr>
<tr>
<td>Corresponding to ...</td>
<td>DB31, ... DBB86 - DBB87 (M function for spindle), axis-specific</td>
</tr>
<tr>
<td></td>
<td>DB21, ... DBB58, DBB68 - DBB97 (M function for spindle), channel-specific</td>
</tr>
</tbody>
</table>
An S function was output from the NCK to the PLC. The output occurs by means of: See "Corresponds to ..." below

The following S functions are output here:
- \( S \ldots \) as spindle speed in rpm (programmed value)
- \( S \ldots \) as constant cutting rate in m/min or ft/min

The following S functions are not output here:
- \( S \ldots \) as the programmed spindle speed limiting \( G25 \)
- \( S \ldots \) as the programmed spindle speed limiting \( G26 \)
- \( S \ldots \) as the spindle speed in rpm if a spindle was not defined in the controller
- \( S \ldots \) as the dwell time in spindle revolutions

Corresponding to ...
- \( DB31, \ldots DBB88 - DBB91 (S function for spindle), axis-specific \)
- \( DB21, \ldots DBB60, DBB98 - DBB115 (S function for spindle), channel-specific \)

### Spindle with SMI 24 (Weiss spindle)

<table>
<thead>
<tr>
<th>DB31, ... DBX132.0</th>
<th>Sensors available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Power-up</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>The sensor required for spindles with SMI 24 is available.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>The sensor required for spindles with SMI 24 is not available.</td>
</tr>
</tbody>
</table>

Corresponding to ...
- \( DB31, \ldots DBX132.1: Sensor S1 (clamped state) is available \)
- \( DB31, \ldots DBX132.4: Sensor S4 is available \)
- \( DB31, \ldots DBX132.5: Sensor S5 is available \)

<table>
<thead>
<tr>
<th>DB31, ... DBX132.1</th>
<th>Sensor S1 available (clamped state)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Power-up</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>Sensor S1 is available.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>Sensor S1 is not available.</td>
</tr>
</tbody>
</table>

Corresponding to ...
- \( DB31, \ldots DBW134 (status of the clamping system) \)
- \( DB31, \ldots DBW136 (analog value: clamped state) \)

<table>
<thead>
<tr>
<th>DB31, ... DBX132.4</th>
<th>Sensor S4 available (piston end position)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Power-up</td>
</tr>
<tr>
<td>Signal state 1</td>
<td>Sensor S4 is available.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>Sensor S4 is not available.</td>
</tr>
</tbody>
</table>

Corresponding to ...
- \( DB31, \ldots DBX138.4 (Sensor S4: piston end position) \)
Sensor S5 available (angular position of the motor shaft)

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Power-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>Sensor S5 is available.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>Sensor S5 is not available.</td>
</tr>
</tbody>
</table>

Corresponding to ... DB31, … DBX138.5 (Sensor S5: angular position of the motor shaft)

State value is generated, speed limitation p5043 is active

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Power-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>The state value is generated and the speed limitations from drive parameter p5043 are active.</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>The state value is not generated and the speed limitations from drive parameter p5043 are not active.</td>
</tr>
</tbody>
</table>

Note
When generating the state value, the analog voltage values of sensor S1 are transformed into discrete state values of drive parameter r5001.

Corresponding to ... DB31, … DBX134 (clamped state)
Drive parameters: r5001
System variable: $VA_MOT_CLAMPING_STATE[<axis>]
OPI variables: vaMotClampingState

Status of the clamping system (sensor S1)

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
</table>

Depending on the position of the clamping device, sensor S1 supplies an analog voltage value. To simplify the evaluation of the clamped state, the analog voltage of sensor module SMI 24 is converted into a state value.

The state values correspond to certain voltage ranges. The voltage ranges can be set via: Drive parameter p5041[0...5].

<table>
<thead>
<tr>
<th>State value</th>
<th>Clamped state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sensor S1 not available or state values inactive</td>
</tr>
<tr>
<td>1</td>
<td>State initialization running</td>
</tr>
<tr>
<td>2</td>
<td>Released with signal (error state)</td>
</tr>
<tr>
<td>3</td>
<td>Released</td>
</tr>
<tr>
<td>4</td>
<td>Clamping with tool</td>
</tr>
<tr>
<td>5</td>
<td>Releasing with tool</td>
</tr>
<tr>
<td>6</td>
<td>Releasing without tool</td>
</tr>
<tr>
<td>7</td>
<td>Clamped with tool AND S4 == 0</td>
</tr>
<tr>
<td>8</td>
<td>Clamped with tool AND S4 == 1</td>
</tr>
<tr>
<td>9</td>
<td>Clamping without tool</td>
</tr>
<tr>
<td>10</td>
<td>Clamped without tool</td>
</tr>
<tr>
<td>11</td>
<td>Clamped with signal (error state)</td>
</tr>
</tbody>
</table>

Corresponding to ... DB31, … DBW136 (analog value: clamped state)
Drive parameters: p5041[0...5], p5043[0...6]
### DB31, ... DBW136  
**Analog measured value: Of the clamping system**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor S1 supplies an analog voltage value: 0 - 10 V. The analog value of the clamped state is mapped to: 0 - 10000 increments, resolution 1 mV</td>
<td></td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
<tr>
<td>SIMATIC S7 input module: 0 - 27648 increments, resolution 0.36 mV</td>
<td></td>
</tr>
<tr>
<td>Adaptation factor if you change to a spindle with SMI 24: 2,7648</td>
<td></td>
</tr>
<tr>
<td>Corresponding to ... DB31, ... DBW134 (clamped state)</td>
<td></td>
</tr>
<tr>
<td>Drive parameters: p5041[0...5], p5043[0...6]</td>
<td></td>
</tr>
</tbody>
</table>

### DB31, ... DBX138.4  
**Sensor S4, piston end position**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>The piston is in position, i.e. the piston is free to move</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>The piston is not in position</td>
</tr>
<tr>
<td>Corresponding to ... DB31, ... DBX132.4 (Sensor S4 available)</td>
<td></td>
</tr>
</tbody>
</table>

### DB31, ... DBX138.5  
**Sensor S5, angular position of the motor shaft**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1</td>
<td>The motor shaft is in position (requirement: The spindle is stationary)</td>
</tr>
<tr>
<td>Signal state 0</td>
<td>The motor shaft is not aligned</td>
</tr>
<tr>
<td>Corresponding to ... DB31, ... DBX132.5 (sensor S5 available)</td>
<td></td>
</tr>
</tbody>
</table>
19.12 Feeds (V1)

19.12.1 Signals to channel (DB21, ...)

<table>
<thead>
<tr>
<th>DB21, ... DBX0.6</th>
<th>Activate dry run feed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal state 0 or edge change</strong> 0 → 1</td>
<td>The dry run feedrate defined using the setting data: SD42100 $SC_DRY_RUN_FEED is used instead of the programmed feed (for G01, G02, G03) if the dry run feedrate is greater than that programmed. The dry run feedrate is effective after the reset state. This interface signal is evaluated at NC start when the channel is in the &quot;reset&quot; state. The dry run feed can be activated from the PLC or operator panel. When selected from the operator panel front, the PLC interface signal: DB21, ... DBX24.6 (dry run feed selected) is set and transferred from the PLC basic program to the interface signal: DB21, ... DBX0.6 (activate dry run feed). When selected using the PLC, the IS &quot;activate dry run feed&quot; should be set from the PLC user program.</td>
</tr>
</tbody>
</table>

| Signal state 0 or edge change 1 → 0 | The programmed feed is used. Effective after the reset state. |

**Application example(s)** Testing a workpiece program with an increased feedrate.

**Special cases, errors, ....** If the signal changes to "0" within a G33 block, the programmed feed is not activated until the end of the block is reached, since an NC stop was not triggered.

**Corresponding to ...** DB21, ... DBX24.6 (dry run feedrate selected) SD42100 $SC_DRY_RUN_FEED (dry run feedrate)
<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>Feedrate override</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBB4</td>
<td></td>
</tr>
</tbody>
</table>

Edge evaluation: No  
Signal(s) updated: Cyclically
### Feedrate override

The feedrate override can be defined via the PLC in binary or Gray coding. With binary coding, the feed value is interpreted in %. 0% to 200% feed changes are possible, in accordance with the binary value in the byte. The following permanent assignment applies:

<table>
<thead>
<tr>
<th>Code</th>
<th>Feed rate override factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0.00 ± 0%</td>
</tr>
<tr>
<td>00000001</td>
<td>0.01 ± 1%</td>
</tr>
<tr>
<td>00000010</td>
<td>0.02 ± 2%</td>
</tr>
<tr>
<td>00000011</td>
<td>0.03 ± 3%</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>11001000</td>
<td>2.00 ± 200%</td>
</tr>
</tbody>
</table>

Binary values > 200 are limited to 200%.

The machine data: MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary-coded override switch) can be used to additionally limit the maximum feedrate override.

In gray coding, the following codes are assigned to the individual switch settings:
### Feedrate override

<table>
<thead>
<tr>
<th>Switch position</th>
<th>Code</th>
<th>Feed rate override factor (standard values)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00001</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>00011</td>
<td>0.01</td>
</tr>
<tr>
<td>3</td>
<td>00010</td>
<td>0.02</td>
</tr>
<tr>
<td>4</td>
<td>00110</td>
<td>0.04</td>
</tr>
<tr>
<td>5</td>
<td>00111</td>
<td>0.06</td>
</tr>
<tr>
<td>6</td>
<td>00101</td>
<td>0.08</td>
</tr>
<tr>
<td>7</td>
<td>00100</td>
<td>0.10</td>
</tr>
<tr>
<td>8</td>
<td>01100</td>
<td>0.20</td>
</tr>
<tr>
<td>9</td>
<td>01101</td>
<td>0.30</td>
</tr>
<tr>
<td>10</td>
<td>01111</td>
<td>0.40</td>
</tr>
<tr>
<td>11</td>
<td>01110</td>
<td>0.50</td>
</tr>
<tr>
<td>12</td>
<td>01010</td>
<td>0.60</td>
</tr>
<tr>
<td>13</td>
<td>01011</td>
<td>0.70</td>
</tr>
<tr>
<td>14</td>
<td>01001</td>
<td>0.75</td>
</tr>
<tr>
<td>15</td>
<td>01000</td>
<td>0.80</td>
</tr>
<tr>
<td>16</td>
<td>11000</td>
<td>0.85</td>
</tr>
<tr>
<td>17</td>
<td>11001</td>
<td>0.90</td>
</tr>
<tr>
<td>18</td>
<td>11011</td>
<td>0.95</td>
</tr>
<tr>
<td>19</td>
<td>11010</td>
<td>1.00</td>
</tr>
<tr>
<td>20</td>
<td>11110</td>
<td>1.05</td>
</tr>
<tr>
<td>21</td>
<td>11111</td>
<td>1.10</td>
</tr>
<tr>
<td>22</td>
<td>11101</td>
<td>1.15</td>
</tr>
<tr>
<td>23</td>
<td>11100</td>
<td>1.20</td>
</tr>
<tr>
<td>24</td>
<td>10100</td>
<td>1.20</td>
</tr>
<tr>
<td>25</td>
<td>10101</td>
<td>1.20</td>
</tr>
<tr>
<td>26</td>
<td>10111</td>
<td>1.20</td>
</tr>
<tr>
<td>27</td>
<td>10110</td>
<td>1.20</td>
</tr>
<tr>
<td>28</td>
<td>10010</td>
<td>1.20</td>
</tr>
<tr>
<td>29</td>
<td>10011</td>
<td>1.20</td>
</tr>
<tr>
<td>30</td>
<td>10001</td>
<td>1.20</td>
</tr>
<tr>
<td>31</td>
<td>10000</td>
<td>1.20</td>
</tr>
</tbody>
</table>

The factors listed in the table for the feedrate override are stored in the machine data: MD12030 $MN_OVR_FACTOR_FEEDRATE [n].

The table contains the default settings.

The number of possible switch settings for standard machine panels is described in the Configuration Manual for SINUMERIK 840D.

**Corresponding to ...**

- **DB21, ... DBX6.7 (feedrate override active)**
- **MD12030 $MN_OVR_FACTOR_FEEDRATE [n] (evaluation of the path feedrate override switch)**
- **MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary coded override switch)**
### Z1: NC/PLC interface signals

#### 19.12 Feeds (V1)

<table>
<thead>
<tr>
<th>DB21, ...</th>
<th>DBB5</th>
<th>Rapid traverse override</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
<td></td>
</tr>
</tbody>
</table>

---

Basic Functions

Function Manual, 01/2015, 6FC5397-0BP40-5BA2

1656
The rapid traverse override can be entered via the PLC in either the binary or Gray code. For binary coding, the rapid traverse override is interpreted as a %. 0% to 100% feed changes are possible, in accordance with the binary value in the byte. The following permanent assignment applies:

<table>
<thead>
<tr>
<th>Code</th>
<th>Rapid traverse override factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0.00 ± 0%</td>
</tr>
<tr>
<td>00000001</td>
<td>0.01 ± 1%</td>
</tr>
<tr>
<td>00000010</td>
<td>0.02 ± 2%</td>
</tr>
<tr>
<td>00000011</td>
<td>0.03 ± 3%</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>11001000</td>
<td>1.00 ± 100%</td>
</tr>
</tbody>
</table>

Binary values > 100 are limited to 100%.
Using the machine data: MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary-coded override switch), the maximum rapid traverse override can be additionally limited.

In gray coding, the following codes are assigned to the individual switch settings:
### Rapid traverse override

The factors listed in the table for the rapid traverse override are stored in the machine data: MD12050 $MN_OVR_FACTOR_RAPID_TRA[n].

The table contains the default settings.

The number of possible switch settings for standard machine panels is described in the Configuration Manual for SINUMERIK 840D.

**Corresponding to** [DB21, DBX6.6 (rapid traverse override active)]

<table>
<thead>
<tr>
<th>Switch position</th>
<th>Code</th>
<th>Rapid traverse override factor (standard values)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00001</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>00011</td>
<td>0.01</td>
</tr>
<tr>
<td>3</td>
<td>00010</td>
<td>0.02</td>
</tr>
<tr>
<td>4</td>
<td>00110</td>
<td>0.04</td>
</tr>
<tr>
<td>5</td>
<td>00111</td>
<td>0.06</td>
</tr>
<tr>
<td>6</td>
<td>00101</td>
<td>0.08</td>
</tr>
<tr>
<td>7</td>
<td>00100</td>
<td>0.10</td>
</tr>
<tr>
<td>8</td>
<td>01100</td>
<td>0.20</td>
</tr>
<tr>
<td>9</td>
<td>01101</td>
<td>0.30</td>
</tr>
<tr>
<td>10</td>
<td>01111</td>
<td>0.40</td>
</tr>
<tr>
<td>11</td>
<td>01110</td>
<td>0.50</td>
</tr>
<tr>
<td>12</td>
<td>01010</td>
<td>0.60</td>
</tr>
<tr>
<td>13</td>
<td>01011</td>
<td>0.70</td>
</tr>
<tr>
<td>14</td>
<td>01001</td>
<td>0.75</td>
</tr>
<tr>
<td>15</td>
<td>01000</td>
<td>0.80</td>
</tr>
<tr>
<td>16</td>
<td>11000</td>
<td>0.85</td>
</tr>
<tr>
<td>17</td>
<td>11001</td>
<td>0.90</td>
</tr>
<tr>
<td>18</td>
<td>11011</td>
<td>0.95</td>
</tr>
<tr>
<td>19</td>
<td>11010</td>
<td>1.00</td>
</tr>
<tr>
<td>20</td>
<td>11110</td>
<td>1.00</td>
</tr>
<tr>
<td>21</td>
<td>11111</td>
<td>1.00</td>
</tr>
<tr>
<td>22</td>
<td>11101</td>
<td>1.00</td>
</tr>
<tr>
<td>23</td>
<td>11100</td>
<td>1.00</td>
</tr>
<tr>
<td>24</td>
<td>10100</td>
<td>1.00</td>
</tr>
<tr>
<td>25</td>
<td>10101</td>
<td>1.00</td>
</tr>
<tr>
<td>26</td>
<td>10111</td>
<td>1.00</td>
</tr>
<tr>
<td>27</td>
<td>10110</td>
<td>1.00</td>
</tr>
<tr>
<td>28</td>
<td>10010</td>
<td>1.00</td>
</tr>
<tr>
<td>29</td>
<td>10011</td>
<td>1.00</td>
</tr>
<tr>
<td>30</td>
<td>10001</td>
<td>1.00</td>
</tr>
<tr>
<td>31</td>
<td>10000</td>
<td>1.00</td>
</tr>
</tbody>
</table>
### Feed disable

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>The signal is active in one channel in all operating modes. The signal disables the feed for all of the axes (geometry and synchronized) that interpolate relative to one another as long as G33 (thread) is not active. All axes are brought to a standstill but still maintaining the path contour. When the feed disable is canceled (0 signal), the interrupted part program is continued. The signal triggers a feed disable for all positioning axes. This signal brings all traversing axes to a standstill with controlled braking (ramp stop). No alarm is output. The position control is retained, i.e. the following error is eliminated. If a travel request is issued for an axis with an active &quot;Feed disable&quot;, then this is kept. The queued travel request is executed immediately when the &quot;Feed disable&quot; is canceled. If the axis is interpolating in relation to others, this also applies to these axes.</td>
</tr>
<tr>
<td>Signal state 0 or edge change 1 → 0</td>
<td>The feedrate is enabled for all axes of the channel. If a travel request (&quot;travel command&quot;) exists for an axis or group of axes when the &quot;feed disable&quot; is canceled, then this is executed immediately.</td>
</tr>
</tbody>
</table>

**Application example(s)**
- Stopping machining by selecting FEED OFF on the machine control panel.

**Special cases, errors, ......**
- The feed disable is inactive when G33 is active.

### Rapid traverse override active

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
</table>
| Signal state 1 or edge change 0 → 1 | The rapid traverse override between 0 and a maximum of 100% entered in the PLC interface is channel-specific. The override factor is defined using the machine data:
- MD12040 $MN_OVR_RAPID_IS_GRAY_CODE (rapid traverse override switch gray coded)
- MD12050 $MN_OVR_FACTOR_RAPID_TRA [n] (evaluation of the rapid traverse override switch). |
| Signal state 0 or edge change 1 → 0 | The rapid traverse override entered at the PLC interface is ignored. When the rapid traverse override is inactive, the NC always uses 100% as the internal override factor. Exceptions are the zero setting for a binary interface and the 1st switch setting for a Gray-coded interface. In these cases, the override factors entered at the PLC interface are used. With a binary interface, the override factor = 0. With a gray-coded interface, the value entered in the machine data for the 1st switch setting is output as the override value. |

**Application example(s)**
- The override value is generally selected using the rapid traverse override switch on the machine control panel.
- Using the interface signal:
  - DB21, ... DBX6.6 (rapid traverse override active), the rapid traverse override switch can be enabled from the PLC user program while commissioning a new NC program, e.g. using the key-operated switch.

**Special cases, errors, ......**
- The rapid traverse override is inactive when G33, G63, G331, G332 are active.

**Corresponding to ...**
- DB21, ... DBB5 (rapid traverse override)
### 19.12 Feeds (V1)

#### Z1: NC/PLC interface signals

<table>
<thead>
<tr>
<th>DB21, ... DB6X8.7</th>
<th>Feedrate override active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>The feedrate override between 0 and a maximum of 200% entered at the PLC interface is active for the path feedrate and therefore automatically for the related axes. In JOG mode, the feedrate override acts directly on the axes. The override factor is entered using the machine data: MD12020 $MN_OVR_FEED_IS_GRAY_CODE (path feedrate override factor, gray-coded) and MD12030 $MN_OVR_FACTOR_FEEDRATE [n] (evaluation of the path feedrate override switch)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB21, ... DBX12.3, DBX16.3, DBX20.3</th>
<th>Feed stop (Geometry axis 1 to 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: No</td>
<td>Signal(s) updated: Cyclically</td>
</tr>
<tr>
<td><strong>Signal state 1 or edge change 0 → 1</strong></td>
<td>The signal is only active in JOG mode. The signal stops the feed of the geometry axis. This signal brings all traversing axes to a standstill with controlled braking (ramp stop). No alarm is output. The position control is retained, i.e. the following error is eliminated. If, for a geometry axis, a travel request is issued with an active &quot;feed stop&quot;, the request is kept. This queued travel request is executed immediately after the &quot;feed stop&quot; is canceled.</td>
</tr>
<tr>
<td><strong>Signal state 0 or edge change 1 → 0</strong></td>
<td>The feed is enabled for the geometry axis. If, for the geometry axis, a travel request (&quot;travel command&quot;) is active when the &quot;feed stop&quot; is canceled, this is executed immediately.</td>
</tr>
</tbody>
</table>
### Dry run feedrate selected

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal state 1 or edge change</strong> 0 → 1</td>
<td><strong>Dry run feedrate is selected.</strong>&lt;br&gt;Instead of the programmed feedrate, the dry run feedrate entered in setting data:&lt;br&gt;SD42100 $SC_DRY_RUN_FEED is active.&lt;br&gt;When activated from the operator panel, the dry run feed signal is automatically entered in the PLC interface and transmitted by the PLC basic program to the PLC interface signal:&lt;br&gt;DB21, ... DBX0.6 (active dry run feed).</td>
</tr>
<tr>
<td><strong>Signal state 0 or edge change</strong> 1 → 0</td>
<td><strong>Dry run feedrate is not selected.</strong>&lt;br&gt;The programmed feedrate is active.</td>
</tr>
<tr>
<td><strong>Corresponding to</strong></td>
<td><strong>DB21, ... DBX0.6 (activate dry run feed)</strong>&lt;br&gt;SD42100 $SC_DRY_RUN_FEED (dry run feedrate)</td>
</tr>
</tbody>
</table>

### Feedrate override selected for rapid traverse

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal state 1 or edge change</strong> 0 → 1</td>
<td><strong>The feedrate override switch should also be active as rapid traverse override switch.</strong>&lt;br&gt;Override values above 100% are limited to the maximum value for 100% rapid traverse override.&lt;br&gt;The interface signal:&lt;br&gt;DB21, ... DBX25.3 (feedrate override for rapid traverse selected) is automatically entered into the PLC interface from the operator panel and transferred from the PLC basic program to the PLC interface signal:&lt;br&gt;DB21, ... DBX6.6 (rapid traverse override active).&lt;br&gt;Further, the interface signal:&lt;br&gt;DB21, ... DBB4 (feedrate override) is copied from the PLC basic program to the interface signal:&lt;br&gt;DB21, ... DBB5 (rapid traverse override).</td>
</tr>
<tr>
<td><strong>Signal state 0 or edge change</strong> 1 → 0</td>
<td><strong>The feedrate override switch should not be activated as rapid traverse override switch.</strong></td>
</tr>
<tr>
<td><strong>Application example(s)</strong></td>
<td><strong>The signal is used when no separate rapid traverse override switch is available.</strong></td>
</tr>
</tbody>
</table>
**Activate fixed feedrate 1 - 4 for path/geometry axes**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
</table>

**Description**

These signals are used to select/de-select the function "fixed feed" and define which fixed feed should be effective for path/geometry axes.

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Fixed feed is de-selected</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Fixed feed 1 is selected</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Fixed feed 2 is selected</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Fixed feed 3 is selected</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Fixed feed 4 is selected</td>
</tr>
</tbody>
</table>

**Corresponding to ...**

- MD12202 $MN_PERMANENT_FEED[n]
- MD12200 $MN_RUN_OVERRIDE_0
### 19.12.2 Signals to axis/spindle (DB31, ...)

<table>
<thead>
<tr>
<th>DB31, ...</th>
<th>Feedrate override (axis-specific)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBB0</td>
<td></td>
</tr>
</tbody>
</table>

Edge evaluation: No  
Signal(s) updated: Cyclically
Feedrate override (axis-specific)

The axis-specific feedrate override can be defined via the PLC in binary or Gray coding. With binary coding, the feed value is interpreted in %. 0% to 200% feed changes are possible, in accordance with the binary value in the byte.

The following permanent assignment applies:

<table>
<thead>
<tr>
<th>Code</th>
<th>Axis-specific feed rate override factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0.00 ± 0%</td>
</tr>
<tr>
<td>00000001</td>
<td>0.01 ± 1%</td>
</tr>
<tr>
<td>00000010</td>
<td>0.02 ± 2%</td>
</tr>
<tr>
<td>00000011</td>
<td>0.03 ± 3%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>01100100</td>
<td>1.00 ± 100%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>11001000</td>
<td>2.00 ± 200%</td>
</tr>
</tbody>
</table>

Binary values > 200 are limited to 200%.

Using the machine data:
MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary-coded override switch) the maximum axis-specific feedrate override can be additionally limited.

In gray coding, the following codes are assigned to the individual switch settings:
Feedrate override (axis-specific)

Table: Gray coding for axis-specific feed rate override

<table>
<thead>
<tr>
<th>Switch position</th>
<th>Code</th>
<th>Axial feed rate override factor (standard values)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00001</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>00011</td>
<td>0.01</td>
</tr>
<tr>
<td>3</td>
<td>00010</td>
<td>0.02</td>
</tr>
<tr>
<td>4</td>
<td>00110</td>
<td>0.04</td>
</tr>
<tr>
<td>5</td>
<td>00111</td>
<td>0.06</td>
</tr>
<tr>
<td>6</td>
<td>00101</td>
<td>0.08</td>
</tr>
<tr>
<td>7</td>
<td>00100</td>
<td>0.10</td>
</tr>
<tr>
<td>8</td>
<td>01100</td>
<td>0.20</td>
</tr>
<tr>
<td>9</td>
<td>01101</td>
<td>0.30</td>
</tr>
<tr>
<td>10</td>
<td>01111</td>
<td>0.40</td>
</tr>
<tr>
<td>11</td>
<td>01110</td>
<td>0.50</td>
</tr>
<tr>
<td>12</td>
<td>01010</td>
<td>0.60</td>
</tr>
<tr>
<td>13</td>
<td>01011</td>
<td>0.70</td>
</tr>
<tr>
<td>14</td>
<td>01001</td>
<td>0.75</td>
</tr>
<tr>
<td>15</td>
<td>01000</td>
<td>0.80</td>
</tr>
<tr>
<td>16</td>
<td>11000</td>
<td>0.85</td>
</tr>
<tr>
<td>17</td>
<td>11001</td>
<td>0.90</td>
</tr>
<tr>
<td>18</td>
<td>11011</td>
<td>0.95</td>
</tr>
<tr>
<td>19</td>
<td>11010</td>
<td>1.00</td>
</tr>
<tr>
<td>20</td>
<td>11110</td>
<td>1.05</td>
</tr>
<tr>
<td>21</td>
<td>11111</td>
<td>1.10</td>
</tr>
<tr>
<td>22</td>
<td>11101</td>
<td>1.15</td>
</tr>
<tr>
<td>23</td>
<td>11100</td>
<td>1.20</td>
</tr>
<tr>
<td>24</td>
<td>10100</td>
<td>1.20</td>
</tr>
<tr>
<td>25</td>
<td>10101</td>
<td>1.20</td>
</tr>
<tr>
<td>26</td>
<td>10111</td>
<td>1.20</td>
</tr>
<tr>
<td>27</td>
<td>10110</td>
<td>1.20</td>
</tr>
<tr>
<td>28</td>
<td>10010</td>
<td>1.20</td>
</tr>
<tr>
<td>29</td>
<td>10011</td>
<td>1.20</td>
</tr>
<tr>
<td>30</td>
<td>10001</td>
<td>1.20</td>
</tr>
<tr>
<td>31</td>
<td>10000</td>
<td>1.20</td>
</tr>
</tbody>
</table>

The factors listed in the table for the axial feedrate override are stored in the NC-specific machine data:
MD12010 $MN_OVR_FACTOR_AX_SPEED [n]

The table contains the default settings.
The number of possible switch settings for standard machine panels is described in the Configuration Manual for SINUMERIK 840D.

Corresponding to ...
MD12010 $MN_OVR_FACTOR_AX_SPEED [n] (evaluation of the axis feedrate override switch)
MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary coded override switch)
### Override active

<table>
<thead>
<tr>
<th>Signal state 1 or edge change 0 → 1</th>
<th>Feedrate override active:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The axis-specific feedrate override between 0 and a maximum of 200% entered in the PLC interface is used.</td>
</tr>
<tr>
<td></td>
<td>The override factor is defined using the machine data:</td>
</tr>
<tr>
<td></td>
<td>MD12000 $MN_OVR_AX_IS_GRAY_CODE (axis feedrate override switch gray coded)</td>
</tr>
<tr>
<td></td>
<td>MD12010 $MN_OVR_FACTOR_AX_SPEED [n] (evaluation of the axis feedrate override switch).</td>
</tr>
</tbody>
</table>

**Spindle override active:**

The spindle override - input at the PLC interface - of 0 to a maximum of 200% is taken into account.

The override factor is entered using the machine data:

- MD12060 $MN_OVR_SPIND_IS_GRAY_CODE (spindle override switch, Gray coded)
- MD12070 $MN_OVR_FACTOR_SPIND_SPEED [n] (evaluation of the spindle override switch).

<table>
<thead>
<tr>
<th>Signal state 0 or edge change 1 → 0</th>
<th>The existing axis-specific feedrate override or spindle override is not active.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the feedrate override is inactive, &quot;100%&quot; is used as the internal override factor.</td>
</tr>
<tr>
<td></td>
<td>Exceptions are the zero setting for a binary interface and the 1st switch setting for a Gray-coded interface. In these cases, the override factors entered at the PLC interface are used.</td>
</tr>
<tr>
<td></td>
<td>With a binary interface, the override factor = 0. With a gray-coded interface, the value entered in the machine data for the 1st switch setting is output as the override value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application example(s)</th>
<th>The override value is generally specified using the axis-specific feedrate override switch or the spindle override switch on the machine control panel.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The &quot;feedrate override active&quot; signal can be used to enable the feedrate override switch from the PLC user program, e.g. using the key-operated switch when commissioning a new NC program.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Special cases, errors, ......</th>
<th>The spindle override is always accepted with 100% in the spindle &quot;Oscillation mode&quot;.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The spindle override acts on the programmed values before the limits (e.g. G26, LIMS...) intervene.</td>
</tr>
<tr>
<td></td>
<td>The feedrate override is ineffective for:</td>
</tr>
<tr>
<td></td>
<td>- Active G33</td>
</tr>
<tr>
<td></td>
<td>- Active G63 (the override is defined in the NC at 100%)</td>
</tr>
<tr>
<td></td>
<td>- Active G331, G332 (the override is defined in the NC at 100%)</td>
</tr>
<tr>
<td></td>
<td>The spindle override is inactive for:</td>
</tr>
<tr>
<td></td>
<td>- Active G63 (the override is defined in the NC at 100%)</td>
</tr>
</tbody>
</table>

| Corresponding to ... | DB31, ..., DBX1.7 (feed/spindle override) |
**DB31, ... DBX3.2 - DBX3.5**

### Activate fixed feedrate 1 - 4 for machine axes

**Edge evaluation:** No  
**Signal(s) updated:** Cyclically

**Description**

These signals are used to select/de-select the function “fixed feed” and define which fixed feed should be effective for machine axes.

<table>
<thead>
<tr>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Fixed feed is de-selected</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Fixed feed 1 is selected</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Fixed feed 2 is selected</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Fixed feed 3 is selected</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Fixed feed 4 is selected</td>
</tr>
</tbody>
</table>

**Corresponding to ...**

- MD12202 $MN_PERMANENT_FEED\[n\] 
- MD12200 $MN_RUN_OVERRIDE_0

---

**DB31, ... DBX4.3**

### Feed stop / spindle stop (axis-specific)

**Edge evaluation:** No  
**Signal(s) updated:** Cyclically

**Signal state 1 or edge change 0 → 1**

The signal is active in all modes.

**Feed stop:**

The signal triggers a "feed stop" for the axis. This signal brings all traversing axes to a standstill with controlled braking (ramp stop). No alarm is output.

The signal triggers a "feed stop" for all path axes interpolating relative to each other when the "feed stop" is activated for any one of these path axes. In this case, all the axes are brought to a stop with adherence to the path contour. When the "feed stop" signal is canceled, execution of the interrupted part program is resumed.

The position control is retained, i.e. the following error is eliminated.

If a travel request is issued for an axis with an active "feed stop", this is kept. This queued travel request is executed immediately after the "feed stop" is canceled.

If the axis is interpolating in relation to others, this also applies to these axes.

**Spindle stop:**

The spindle is brought to a standstill along the acceleration characteristic.

In the positioning mode, when the "Spindle stop" signal is set positioning is interrupted. The above response applies with respect to individual axes.

**Signal state 0 or edge change 1 → 0**

**Feed stop:**

The feedrate is enabled for the axis.

If a travel request ("travel command") is active when the "feed stop" is canceled, this is executed immediately.

**Spindle stop:**

The speed is enabled for the spindle.

The spindle is accelerated to the previous speed setpoint with the acceleration characteristic or, in positioning mode, positioning is resumed.
**Feed stop / spindle stop (axis-specific)**

<table>
<thead>
<tr>
<th>Application example(s)</th>
<th>Feed stop:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The traversing motion of the machine axes is not started with &quot;feed stop&quot;, if, for example, certain operating states exist at the machine that do not permit the axes to be moved (e.g. a door is not closed).</td>
</tr>
<tr>
<td><strong>Spindle stop:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>In order to change a tool</td>
</tr>
<tr>
<td></td>
<td>To enter help functions (M, S, H, T, D and F functions) during setup.</td>
</tr>
</tbody>
</table>

| Special cases, errors, ...... | Spindle stop is inactive when G331, G332 are active. |

**Spindle override**

<table>
<thead>
<tr>
<th>Edge evaluation: No</th>
<th>Signal(s) updated: Cyclically</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>The spindle override can be defined via the PLC in binary or Gray coding.</td>
</tr>
<tr>
<td></td>
<td>The override value determines the percentage of the programmed speed setpoint that is issued to the spindle.</td>
</tr>
<tr>
<td></td>
<td>With binary coding, the override is interpreted in %. 0% to 200% feed changes are possible, in accordance with the binary value in the byte.</td>
</tr>
<tr>
<td></td>
<td>The following permanent assignment applies:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>Spindle override factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0.00 ± 0%</td>
</tr>
<tr>
<td>00000001</td>
<td>0.01 ± 1%</td>
</tr>
<tr>
<td>00000010</td>
<td>0.02 ± 2%</td>
</tr>
<tr>
<td>00000011</td>
<td>0.03 ± 3%</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>01100100</td>
<td>1.00 ± 100%</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>11001000</td>
<td>2.00 ± 200%</td>
</tr>
</tbody>
</table>

Binary values > 200 are limited to 200%.

The machine data: MD12100 $MN_OVR_FACTOR_LIMIT_BIN (limit for binary-coded override switch) can be used to additionally limit the maximum spindle override.
Spindle override

Signal state 1 or edge change 0 → 1

In gray coding, the following codes are assigned to the individual switch settings:

<table>
<thead>
<tr>
<th>Switch position</th>
<th>Code</th>
<th>Spindle override factor (standard values)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00001</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>00011</td>
<td>0.55</td>
</tr>
<tr>
<td>3</td>
<td>00010</td>
<td>0.60</td>
</tr>
<tr>
<td>4</td>
<td>00110</td>
<td>0.65</td>
</tr>
<tr>
<td>5</td>
<td>01111</td>
<td>0.70</td>
</tr>
<tr>
<td>6</td>
<td>00101</td>
<td>0.75</td>
</tr>
<tr>
<td>7</td>
<td>00100</td>
<td>0.80</td>
</tr>
<tr>
<td>8</td>
<td>01100</td>
<td>0.85</td>
</tr>
<tr>
<td>9</td>
<td>01101</td>
<td>0.90</td>
</tr>
<tr>
<td>10</td>
<td>01111</td>
<td>0.95</td>
</tr>
<tr>
<td>11</td>
<td>01110</td>
<td>1.00</td>
</tr>
<tr>
<td>12</td>
<td>01010</td>
<td>1.05</td>
</tr>
<tr>
<td>13</td>
<td>01011</td>
<td>1.10</td>
</tr>
<tr>
<td>14</td>
<td>01001</td>
<td>1.15</td>
</tr>
<tr>
<td>15</td>
<td>01000</td>
<td>1.20</td>
</tr>
<tr>
<td>16</td>
<td>11000</td>
<td>1.20</td>
</tr>
<tr>
<td>17</td>
<td>11001</td>
<td>1.20</td>
</tr>
<tr>
<td>18</td>
<td>11011</td>
<td>1.20</td>
</tr>
<tr>
<td>19</td>
<td>11010</td>
<td>1.20</td>
</tr>
<tr>
<td>20</td>
<td>11110</td>
<td>1.20</td>
</tr>
<tr>
<td>21</td>
<td>11111</td>
<td>1.20</td>
</tr>
<tr>
<td>22</td>
<td>11101</td>
<td>1.20</td>
</tr>
<tr>
<td>23</td>
<td>11100</td>
<td>1.20</td>
</tr>
<tr>
<td>24</td>
<td>10100</td>
<td>1.20</td>
</tr>
<tr>
<td>25</td>
<td>10101</td>
<td>1.20</td>
</tr>
<tr>
<td>26</td>
<td>10111</td>
<td>1.20</td>
</tr>
<tr>
<td>27</td>
<td>10110</td>
<td>1.20</td>
</tr>
<tr>
<td>28</td>
<td>10010</td>
<td>1.20</td>
</tr>
<tr>
<td>29</td>
<td>10011</td>
<td>1.20</td>
</tr>
<tr>
<td>30</td>
<td>10001</td>
<td>1.20</td>
</tr>
<tr>
<td>31</td>
<td>10000</td>
<td>1.20</td>
</tr>
</tbody>
</table>

The factors listed in the table for spindle override are stored in the machine data:
MD12070 $MN_OVR_FACTOR_SPIIND_SPEED [n]
The table contains the default settings.
The number of possible switch settings for standard machine panels is described in the Configuration Manual for SINUMERIK 840D.

Corresponding to ...

DB31, ... DBX1.7 (override active)
MD12070 $MN_OVR_FACTOR_SPIIND_SPEED [n] (evaluation of the spindle override switch)
MD12100 $MN_FACTOR_LIMIT_BIN (limit for binary-coded override switch)
### 19.12.3 Signals from axis/spindle (DB31, ...)

<table>
<thead>
<tr>
<th>DB31, ... DBX62.2</th>
<th>Revoluional feed rate active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: no</td>
<td>Signal(s) updated: cyclic</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>For programming of G95 (revolutional feed rate) in JOG mode or automatic mode.</td>
</tr>
</tbody>
</table>

**Corresponding to:**
- SD41100 $SN_JOG_REV_IS_ACTIVE (revolutional feed rate for JOG active)
- SD42600 $SC_JOG_FEED_PER_REV_SOURCE (in the JOG mode revolutional feed rate for geometry axes, on which the frame with rotation acts)
- SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE (revolutional feed rate for position axes/spindles)
- MD32040 $MA_JOG_REV_VELO_RAPID (revolutional feed rate for JOG with rapid traverse override)
- MD32050 $MA_JOG_REV_VELO (revolutional feed rate for JOG mode)

<table>
<thead>
<tr>
<th>DB31, ... DBB78 - DBB81</th>
<th>F function for positioning axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge evaluation: no</td>
<td>Signal(s) updated: cyclic</td>
</tr>
<tr>
<td>Signal state 1 or edge change 0 → 1</td>
<td>The F value of a positioning axis programmed in the current block is entered in the axis-specific PLC interface signal. The assignment between the DB number and machine axis number is established using the axis name. The value is retained until it is overwritten by another. Format: Binary number in real format.</td>
</tr>
</tbody>
</table>

**Application example(s):**
- Modification of the programmed F value by the PLC, e.g. by overwriting the selected axis-specific feed rate override.

**Corresponding to:**
- MD22240 $MC_AUXFU_F_SYNC_TYPE (output time F functions)
# A.1 List of abbreviations

<table>
<thead>
<tr>
<th>A</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Output</td>
</tr>
<tr>
<td>ADI4</td>
<td>(Analog drive interface for 4 axes)</td>
</tr>
<tr>
<td>AC</td>
<td>Adaptive Control</td>
</tr>
<tr>
<td>ALM</td>
<td>Active Line Module</td>
</tr>
<tr>
<td>ARM</td>
<td>Rotating induction motor</td>
</tr>
<tr>
<td>AS</td>
<td>Automation system</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange: American coding standard for the exchange of information</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application-Specific Integrated Circuit: User switching circuit</td>
</tr>
<tr>
<td>ASUB</td>
<td>Asynchronous subprogram</td>
</tr>
<tr>
<td>AUXFU</td>
<td>Auxiliary function: Auxiliary function</td>
</tr>
<tr>
<td>STL</td>
<td>Statement List</td>
</tr>
<tr>
<td>UP</td>
<td>User Program</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP</td>
<td>Operating Mode</td>
</tr>
<tr>
<td>BAG</td>
<td>Mode group</td>
</tr>
<tr>
<td>BCD</td>
<td>Binary Coded Decimals: Decimal numbers encoded in binary code</td>
</tr>
<tr>
<td>BERO</td>
<td>Contact-less proximity switch</td>
</tr>
<tr>
<td>BI</td>
<td>Binector Input</td>
</tr>
<tr>
<td>BICO</td>
<td>Binector Connector</td>
</tr>
<tr>
<td>BIN</td>
<td>BI.Nary files: Binary files</td>
</tr>
<tr>
<td>BIOS</td>
<td>Basic Input Output System</td>
</tr>
<tr>
<td>BCS</td>
<td>Basic Coordinate System</td>
</tr>
<tr>
<td>BO</td>
<td>Binector Output</td>
</tr>
<tr>
<td>OPI</td>
<td>Operator Panel Interface</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAD</td>
<td>Computer-Aided Design</td>
</tr>
<tr>
<td>CAM</td>
<td>Computer-Aided Manufacturing</td>
</tr>
<tr>
<td>CC</td>
<td>Compile Cycle: Compile cycles</td>
</tr>
<tr>
<td>CI</td>
<td>Connector Input</td>
</tr>
<tr>
<td>CF Card</td>
<td>Compact Flash Card</td>
</tr>
<tr>
<td>CNC</td>
<td>Computerized Numerical Control: Computer-Supported Numerical Control</td>
</tr>
</tbody>
</table>
## Appendix

### A.1 List of abbreviations

<table>
<thead>
<tr>
<th>C</th>
<th>Connector Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoL</td>
<td>Certificate of License</td>
</tr>
<tr>
<td>COM</td>
<td>Communication</td>
</tr>
<tr>
<td>CPA</td>
<td>Compiler Projecting Data: Configuring data of the compiler</td>
</tr>
<tr>
<td>CRT</td>
<td>Cathode Ray Tube: picture tube</td>
</tr>
<tr>
<td>CSB</td>
<td>Central Service Board: PLC module</td>
</tr>
<tr>
<td>CU</td>
<td>Control Unit</td>
</tr>
<tr>
<td>CP</td>
<td>Communication Processor</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit: Central processing unit</td>
</tr>
<tr>
<td>CR</td>
<td>Carriage Return</td>
</tr>
<tr>
<td>CTS</td>
<td>Clear To Send: Ready to send signal for serial data interfaces</td>
</tr>
<tr>
<td>CUTCOM</td>
<td>Cutter radius Compensation: Tool radius compensation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>Digital-to-Analog Converter</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>Data Block (PLC)</td>
</tr>
<tr>
<td>DBB</td>
<td>Data Block Byte (PLC)</td>
</tr>
<tr>
<td>DBD</td>
<td>Data Block Double word (PLC)</td>
</tr>
<tr>
<td>DBW</td>
<td>Data Block Word (PLC)</td>
</tr>
<tr>
<td>DBX</td>
<td>Data block bit (PLC)</td>
</tr>
<tr>
<td>DDE</td>
<td>Dynamic Data Exchange</td>
</tr>
<tr>
<td>DDS</td>
<td>Drive Data Set: Drive data set</td>
</tr>
<tr>
<td>DIN</td>
<td>Deutsche Industrie Norm</td>
</tr>
<tr>
<td>DIO</td>
<td>Data Input/Output: Data transfer display</td>
</tr>
<tr>
<td>DIR</td>
<td>Directory: Directory</td>
</tr>
<tr>
<td>DLL</td>
<td>Dynamic Link Library</td>
</tr>
<tr>
<td>DO</td>
<td>Drive Object</td>
</tr>
<tr>
<td>DPM</td>
<td>Dual Port Memory</td>
</tr>
<tr>
<td>DPR</td>
<td>Dual Port RAM</td>
</tr>
<tr>
<td>DRAM</td>
<td>Dynamic memory (non-buffered)</td>
</tr>
<tr>
<td>DRF</td>
<td>Differential Resolver Function: Differential revolver function (handwheel)</td>
</tr>
<tr>
<td>DRIVE-CLiQ</td>
<td>Drive Component Link with IQ</td>
</tr>
<tr>
<td>DRY</td>
<td>Dry Run: Dry run feedrate</td>
</tr>
<tr>
<td>DSB</td>
<td>Decoding Single Block: Decoding single block</td>
</tr>
<tr>
<td>DSC</td>
<td>Dynamic Servo Control / Dynamic Stiffness Control</td>
</tr>
<tr>
<td>DW</td>
<td>Data Word</td>
</tr>
<tr>
<td>DWORD</td>
<td>Double Word (currently 32 bits)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>E</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>EES</td>
<td>Execution from External Storage</td>
</tr>
</tbody>
</table>
## A.1 List of abbreviations

<table>
<thead>
<tr>
<th><strong>E</strong></th>
<th><strong>F</strong></th>
<th><strong>G</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O</td>
<td>FB</td>
<td>GC</td>
</tr>
<tr>
<td>ENC</td>
<td>FC</td>
<td>GDIR</td>
</tr>
<tr>
<td>EFP</td>
<td>FEPROM</td>
<td>GEO</td>
</tr>
<tr>
<td>ESD</td>
<td>FIFO</td>
<td>GIA</td>
</tr>
<tr>
<td>EMC</td>
<td>FIPO</td>
<td>GND</td>
</tr>
<tr>
<td>EN</td>
<td>FPU</td>
<td>GP</td>
</tr>
<tr>
<td>ENc</td>
<td>CRC</td>
<td>GS</td>
</tr>
<tr>
<td>EnDat</td>
<td>ESR</td>
<td>GSD</td>
</tr>
<tr>
<td>EPROM</td>
<td>FST</td>
<td>GSDML</td>
</tr>
<tr>
<td>ePS</td>
<td>FBD</td>
<td>GUD</td>
</tr>
<tr>
<td>Network</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EQN</td>
<td>FW</td>
<td></td>
</tr>
<tr>
<td>EN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encoder</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actual value encoder</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encoder: Actual value encoder</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encoder interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Erasable Programmable Read Only Memory: Erasable, electrically programmable read-only memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Services for Internet-based remote machine maintenance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designation for an absolute encoder with 2048 sine signals per revolution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engineering System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extended Stop and Retract</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ETC key &quot;&gt;&quot;; softkey bar extension in the same menu</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Block (PLC)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Call: Function Block (PLC)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flash EPROM: Read and write memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First In First Out: Memory that works without address specification and whose data is read in the same order in which they was stored</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fine interpolator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Floating Point Unit: Floating Point Unit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cutter Radius Compensation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feed Stop: Feedrate stop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Block Diagram (PLC programming method)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firmware</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global Control (PROFIBUS: Broadcast telegram)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global part program memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geometry, e.g. geometry axis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gear Interpolation dAta: Gear interpolation data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal Ground</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic program (PLC)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gear Stage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device master file for describing a PROFIBUS slave</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generic Station Description Markup Language: XML-based description language for creating a GSD file</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global User Data: Global user data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## A.1 List of abbreviations

### H

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEX</td>
<td>Abbreviation for hexadecimal number</td>
</tr>
<tr>
<td>AuxF</td>
<td>Auxiliary function</td>
</tr>
<tr>
<td>HLA</td>
<td>Hydraulic linear drive</td>
</tr>
<tr>
<td>HMI</td>
<td>Human Machine Interface: SINUMERIK user interface</td>
</tr>
<tr>
<td>MSD</td>
<td>Main Spindle Drive</td>
</tr>
<tr>
<td>HW</td>
<td>Hardware</td>
</tr>
</tbody>
</table>

### I

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBN</td>
<td>Commissioning</td>
</tr>
<tr>
<td>ICA</td>
<td>Interpolatory compensation</td>
</tr>
<tr>
<td>IM</td>
<td>Interface Module: Interconnection module</td>
</tr>
<tr>
<td>IMR</td>
<td>Interface Module Receive: Interface module for receiving data</td>
</tr>
<tr>
<td>IMS</td>
<td>Interface Module Send: Interface module for sending data</td>
</tr>
<tr>
<td>INC</td>
<td>Increment: Increment</td>
</tr>
<tr>
<td>INI</td>
<td>Initializing Data: Initializing data</td>
</tr>
<tr>
<td>IPO</td>
<td>Interpolator</td>
</tr>
<tr>
<td>ISA</td>
<td>Industry Standard Architecture</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standardization Organization</td>
</tr>
</tbody>
</table>

### J

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOG</td>
<td>Jogging: Setup mode</td>
</tr>
</tbody>
</table>

### K

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>K_p</td>
<td>Gain factor of control loop</td>
</tr>
<tr>
<td>K_0</td>
<td>Proportional gain</td>
</tr>
<tr>
<td>K_U</td>
<td>Transformation ratio</td>
</tr>
<tr>
<td>LAD</td>
<td>Ladder Diagram (PLC programming method)</td>
</tr>
</tbody>
</table>

### L

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAI</td>
<td>Logic Machine Axis Image: Logical machine axes image</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LCD</td>
<td>Liquid Crystal Display: Liquid crystal display</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode: Light-emitting diode</td>
</tr>
<tr>
<td>LF</td>
<td>Line Feed</td>
</tr>
<tr>
<td>PMS</td>
<td>Position Measuring System</td>
</tr>
<tr>
<td>LR</td>
<td>Position controller</td>
</tr>
<tr>
<td>LSB</td>
<td>Least Significant Bit: Least significant bit</td>
</tr>
<tr>
<td>LUD</td>
<td>Local User Data: User data (local)</td>
</tr>
</tbody>
</table>
## A.1 List of abbreviations

<table>
<thead>
<tr>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>MAIN</td>
<td>Main program: Main program (OB1, PLC)</td>
</tr>
<tr>
<td>MB</td>
<td>Megabyte</td>
</tr>
<tr>
<td>MCI</td>
<td>Motion Control Interface</td>
</tr>
<tr>
<td>MCIS</td>
<td>Motion Control Information System</td>
</tr>
<tr>
<td>MCP</td>
<td>Machine Control Panel: Machine control panel</td>
</tr>
<tr>
<td>MD</td>
<td>Machine Data</td>
</tr>
<tr>
<td>MDA</td>
<td>Manual Data Automatic: Manual input</td>
</tr>
<tr>
<td>MDS</td>
<td>Motor Data Set: Motor data set</td>
</tr>
<tr>
<td>MSGW</td>
<td>Message Word</td>
</tr>
<tr>
<td>MCS</td>
<td>Machine Coordinate System</td>
</tr>
<tr>
<td>MM</td>
<td>Motor Module</td>
</tr>
<tr>
<td>MPF</td>
<td>Main Program File: Main program (NC)</td>
</tr>
<tr>
<td>MCP</td>
<td>Machine control panel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>Numerical Control: Numerical Control</td>
</tr>
<tr>
<td>NCK</td>
<td>Numerical Control Kernel: NC kernel with block preparation, traversing range, etc.</td>
</tr>
<tr>
<td>NCU</td>
<td>Numerical Control Unit: NCK hardware unit</td>
</tr>
<tr>
<td>NRK</td>
<td>Name for the operating system of the NCK</td>
</tr>
<tr>
<td>IS</td>
<td>Interface Signal</td>
</tr>
<tr>
<td>NURBS</td>
<td>Non-Uniform Rational B-Spline</td>
</tr>
<tr>
<td>WO</td>
<td>Work Offset</td>
</tr>
<tr>
<td>NX</td>
<td>Numerical Extension: Axis expansion board</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OB</td>
<td>Organization block in the PLC</td>
</tr>
<tr>
<td>OEM</td>
<td>Original Equipment Manufacturer</td>
</tr>
<tr>
<td>OP</td>
<td>Operator Panel: Operating equipment</td>
</tr>
<tr>
<td>OPI</td>
<td>Operator Panel Interface: Interface for connection to the operator panel</td>
</tr>
<tr>
<td>OPT</td>
<td>Options: Options</td>
</tr>
<tr>
<td>OLP</td>
<td>Optical Link Plug: Fiber optic bus connector</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection: Standard for computer communications</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIQ</td>
<td>Process Image Output</td>
</tr>
<tr>
<td>PII</td>
<td>Process Image Input</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PCIN</td>
<td>Name of the SW for data exchange with the control</td>
</tr>
</tbody>
</table>
### Appendix

#### A.1 List of abbreviations

<table>
<thead>
<tr>
<th>P</th>
<th>Personal Computer Memory Card International Association: Plug-in memory card standardization</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCU</td>
<td>PC Unit: PC box (computer unit)</td>
</tr>
<tr>
<td>PG</td>
<td>Programming device</td>
</tr>
<tr>
<td>PKE</td>
<td>Parameter identification: Part of a PIV</td>
</tr>
<tr>
<td>PIV</td>
<td>Parameter identification: Value (parameterizing part of a PPO)</td>
</tr>
<tr>
<td>PLC</td>
<td>Programmable Logic Control: Adaptation control</td>
</tr>
<tr>
<td>PN</td>
<td>PROFINET</td>
</tr>
<tr>
<td>PNO</td>
<td>PROFIBUS user organization</td>
</tr>
<tr>
<td>PO</td>
<td>POWER ON</td>
</tr>
<tr>
<td>POU</td>
<td>Program Organization Unit</td>
</tr>
<tr>
<td>POS</td>
<td>Position/positioning</td>
</tr>
<tr>
<td>POSMO A</td>
<td>Positioning Motor Actuator: Positioning motor</td>
</tr>
<tr>
<td>POSMO CA</td>
<td>Positioning Motor Compact AC: Complete drive unit with integrated power and control module as well as positioning unit and program memory, AC infeed</td>
</tr>
<tr>
<td>POSMO CD</td>
<td>Positioning Motor Compact DC: Like CA but with DC infeed</td>
</tr>
<tr>
<td>POSMO SI</td>
<td>Positioning Motor Servo Integrated: Positioning motor, DC infeed</td>
</tr>
<tr>
<td>PPO</td>
<td>Parameter Process data Object: Cyclic data telegram for PROFIBUS DP transmission and &quot;Variable speed drives&quot; profile</td>
</tr>
<tr>
<td>PPU</td>
<td>Panel Processing Unit (central hardware for a panel-based CNC, e.g SINUMERIK 828D)</td>
</tr>
<tr>
<td>PROFIBUS</td>
<td>Process Field Bus: Serial data bus</td>
</tr>
<tr>
<td>PRT</td>
<td>Program Test</td>
</tr>
<tr>
<td>PSW</td>
<td>Program control word</td>
</tr>
<tr>
<td>PTP</td>
<td>Point-To-Point: Point-To-Point</td>
</tr>
<tr>
<td>PUD</td>
<td>Program global User Data: Program-global user variables</td>
</tr>
<tr>
<td>PZD</td>
<td>Process data: Process data part of a PPO</td>
</tr>
</tbody>
</table>

| Q   | Quadrant Error Compensation                                                               |

<table>
<thead>
<tr>
<th>R</th>
<th>Random Access Memory: Read/write memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>REF</td>
<td>REFerence point approach function</td>
</tr>
<tr>
<td>REPOS</td>
<td>REPOsition function</td>
</tr>
<tr>
<td>RISC</td>
<td>Reduced Instruction Set Computer: Type of processor with small instruction set and ability to process instructions at high speed</td>
</tr>
<tr>
<td>ROV</td>
<td>Rapid Override: Input correction</td>
</tr>
<tr>
<td>RP</td>
<td>R Parameter, arithmetic parameter, predefined user variable</td>
</tr>
<tr>
<td>RPA</td>
<td>R Parameter Active: Memory area on the NCK for R parameter numbers</td>
</tr>
<tr>
<td>RPY</td>
<td>Roll Pitch Yaw: Rotation type of a coordinate system</td>
</tr>
<tr>
<td>RTLI</td>
<td>Rapid Traverse Linear Interpolation: Linear interpolation during rapid traverse motion</td>
</tr>
</tbody>
</table>
## A.1 List of abbreviations

<table>
<thead>
<tr>
<th>R</th>
<th>Request To Send: Control signal of serial data interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS</td>
<td>Real Time Control Protocol</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Synchronized Action</td>
</tr>
<tr>
<td>SBC</td>
<td>Safe Brake Control: Safe Brake Control</td>
</tr>
<tr>
<td>SBL</td>
<td>Single Block: Single block</td>
</tr>
<tr>
<td>SBR</td>
<td>Subroutine: Subprogram (PLC)</td>
</tr>
<tr>
<td>SD</td>
<td>Setting Data</td>
</tr>
<tr>
<td>SDB</td>
<td>System Data Block</td>
</tr>
<tr>
<td>SEA</td>
<td>Setting Data Active: Identifier (file type) for setting data</td>
</tr>
<tr>
<td>SERUPRO</td>
<td>SEarch RUrn by PROgram test: Search run by program test</td>
</tr>
<tr>
<td>SFB</td>
<td>System Function Block</td>
</tr>
<tr>
<td>SFC</td>
<td>System Function Call</td>
</tr>
<tr>
<td>SGE</td>
<td>Safety-related input</td>
</tr>
<tr>
<td>SGA</td>
<td>Safety-related output</td>
</tr>
<tr>
<td>SH</td>
<td>Safe standstill</td>
</tr>
<tr>
<td>SIM</td>
<td>Single in Line Module</td>
</tr>
<tr>
<td>SK</td>
<td>Softkey</td>
</tr>
<tr>
<td>SKP</td>
<td>Skip: Function for skipping a part program block</td>
</tr>
<tr>
<td>SLM</td>
<td>Synchronous Linear Motor</td>
</tr>
<tr>
<td>SM</td>
<td>Stepper Motor</td>
</tr>
<tr>
<td>SMC</td>
<td>Sensor Module Cabinet Mounted</td>
</tr>
<tr>
<td>SME</td>
<td>Sensor Module Externally Mounted</td>
</tr>
<tr>
<td>SMI</td>
<td>Sensor Module Integrated</td>
</tr>
<tr>
<td>SPF</td>
<td>Sub Routine File: Subprogram (NC)</td>
</tr>
<tr>
<td>PLC</td>
<td>Programmable Logic Controller</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static RAM (non-volatile)</td>
</tr>
<tr>
<td>TNRC</td>
<td>Tool Nose Radius Compensation</td>
</tr>
<tr>
<td>SRM</td>
<td>Synchronous Rotary Motor</td>
</tr>
<tr>
<td>LEC</td>
<td>Leadscrew Error Compensation</td>
</tr>
<tr>
<td>SSI</td>
<td>Serial Synchronous Interface: Synchronous serial interface</td>
</tr>
<tr>
<td>SSL</td>
<td>Block search</td>
</tr>
<tr>
<td>STW</td>
<td>Control word</td>
</tr>
<tr>
<td>GWPS</td>
<td>Grinding Wheel Peripheral Speed</td>
</tr>
<tr>
<td>SW</td>
<td>Software</td>
</tr>
<tr>
<td>SYF</td>
<td>System Files: System files</td>
</tr>
<tr>
<td>SYNACT</td>
<td>SYNchronized ACTion: Synchronized Action</td>
</tr>
</tbody>
</table>
## A.1 List of abbreviations

<table>
<thead>
<tr>
<th>T</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB</td>
<td>Terminal Board (SINAMICS)</td>
</tr>
<tr>
<td>TCP</td>
<td>Tool Center Point: Tool tip</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transport Control Protocol / Internet Protocol</td>
</tr>
<tr>
<td>TCU</td>
<td>Thin Client Unit</td>
</tr>
<tr>
<td>TEA</td>
<td>Testing Data Active: Identifier for machine data</td>
</tr>
<tr>
<td>TIA</td>
<td>Totally Integrated Automation</td>
</tr>
<tr>
<td>TM</td>
<td>Terminal Module (SINAMICS)</td>
</tr>
<tr>
<td>TO</td>
<td>Tool Offset: Tool offset</td>
</tr>
<tr>
<td>TOA</td>
<td>Tool Offset Active: Identifier (file type) for tool offsets</td>
</tr>
<tr>
<td>TRANSMIT</td>
<td>Transform Milling Into Turning: Coordination transformation for milling operations on a lathe</td>
</tr>
<tr>
<td>TTL</td>
<td>Transistor-Transistor Logic (interface type)</td>
</tr>
<tr>
<td>TZ</td>
<td>Technology cycle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UFR</td>
<td>User Frame: Work offset</td>
</tr>
<tr>
<td>SR</td>
<td>Subprogram</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>UPS</td>
<td>Uninterruptible Power Supply</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDI</td>
<td>Internal communication interface between NCK and PLC</td>
</tr>
<tr>
<td>VDI</td>
<td>Verein Deutscher Ingenieure [Association of German Engineers]</td>
</tr>
<tr>
<td>VDE</td>
<td>Verband Deutscher Elektrotechniker [Association of German Electrical Engineers]</td>
</tr>
<tr>
<td>VI</td>
<td>Voltage Input</td>
</tr>
<tr>
<td>VO</td>
<td>Voltage Output</td>
</tr>
<tr>
<td>FDD</td>
<td>Feed Drive</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAR</td>
<td>Smooth Approach and Retraction</td>
</tr>
<tr>
<td>WCS</td>
<td>Workpiece Coordinate System</td>
</tr>
<tr>
<td>T</td>
<td>Tool</td>
</tr>
<tr>
<td>TLC</td>
<td>Tool Length Compensation</td>
</tr>
<tr>
<td>WOP</td>
<td>Workshop-Oriented Programming</td>
</tr>
<tr>
<td>WPD</td>
<td>Workpiece Directory: Workpiece directory</td>
</tr>
<tr>
<td>TRC</td>
<td>Tool Radius Compensation</td>
</tr>
<tr>
<td>TO</td>
<td>Tool Offset</td>
</tr>
<tr>
<td>TM</td>
<td>Tool Management</td>
</tr>
<tr>
<td>TC</td>
<td>Tool change</td>
</tr>
</tbody>
</table>
### A.1 List of abbreviations

<table>
<thead>
<tr>
<th>X</th>
<th>XML</th>
<th>Extensible Markup Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>WOA</td>
<td>Work Offset Active: Identifier for work offsets</td>
</tr>
<tr>
<td></td>
<td>ZSW</td>
<td>Status word (of drive)</td>
</tr>
</tbody>
</table>
A.2 Documentation overview

General documentation

- Sales brochure
  - SINUMERIK 840D sl
  - SINUMERIK 828D
  - SINUMERIK 828D BASIC T
- Catalog NC 62
  - SINUMERIK 840D sl type 1B
- Catalog NC 82
  - SINUMERIK 828D BASIC T/BASIC M
  - SINAMICS S120 combi motors 1FK7 and 1PH8
- Catalog PM 21
  - SIMOTION
  - SINAMICS S120 and motors for production machines
- Configuration Manual
  - EMC design guidelines
- System Manual
  - Ctrl-Energy

User documentation

- Operating Manual
  - Universal
  - Turning
  - Milling
- Programming Manual
  - Basics
  - Job planning
  - Measuring cycles
- Programming Manual
  - ISO turning
  - ISO milling
- Diagnostics Manual
  - Service Manual

Manufacturer / service documentation

- Manual
  - NCU Operation components and networking
- Commissioning Manual
  - Service Manual
  - CNC: NCK, PLC, drive
  - Basic software and operating software
- List Manual
  - Machine data
  - Interface signals
  - Variables
- List Manual
  - Machine data
  - Parameters
  - Variables
- System Manual
  - Guidelines for machine configuration

Manufacturer / service documentation

- Function Manual
  - Basic functions
  - Extended functions
  - Special functions
  - Synchronized actions
  - ISO dialects
- Function Manual
  - Tool management
- Function Manual
  - Drive functions
- Function Manual
  - Safety Integrated
- Function Manual
  - Safety Integrated

Info / training

- Training documents
  - Easy milling with ShopMill
  - Turning made easy with ShopTurn
- Manual
  - Tool and mold making

Electronic documentation

- DOConCD
- My Documentation Manager
- Industry Mall
Glossary

**Absolute dimensions**
A destination for an axis motion is defined by a dimension that refers to the origin of the currently active coordinate system. See → Incremental dimension

**Acceleration with jerk limitation**
In order to optimize the acceleration response of the machine whilst simultaneously protecting the mechanical components, it is possible to switch over in the machining program between abrupt acceleration and continuous (jerk-free) acceleration.

**Address**
An address is the identifier for a certain operand or operand range, e.g. input, output, etc.

**Alarms**
All → messages and alarms are displayed on the operator panel in plain text with date and time and the corresponding symbol for the cancel criterion. Alarms and messages are displayed separately.

1. Alarms and messages in the part program:
   Alarms and messages can be displayed in plain text directly from the part program.

2. Alarms and messages from the PLC:
   Alarms and messages for the machine can be displayed in plain text from the PLC program.
   No additional function block packages are required for this purpose.

**Archiving**
Reading out of files and/or directories on an external memory device.

**Asynchronous subprogram**
Part program that can be started asynchronously to (independently of) the current program status using an interrupt signal (e.g. "Rapid NC input" signal).

**Automatic**
Operating mode of the controller (block sequence operation according to DIN): Operating mode for NC systems in which a → subprogram is selected and executed continuously.
Auxiliary functions

Auxiliary functions enable → part programs to transfer → parameters to the → PLC, which then trigger reactions defined by the machine manufacturer.

Axes

In accordance with their functional scope, the CNC axes are subdivided into:

- Axes: Interpolating path axes
- Auxiliary axes: Non-interpolating feed and positioning axes with an axis-specific feedrate. Auxiliary axes are not involved in actual machining, e.g. tool feeder, tool magazine.

Axis address

See → Axis name

Axis name

To ensure clear identification, all channel and → machine axes of the control system must be designated with unique names in the channel and control system. The → geometry axes are called X, Y, Z. The rotary axes rotating around the geometry axes → are called A, B, C.

Backlash compensation

Compensation for a mechanical machine backlash, e.g. backlash on reversal for ball screws. Backlash compensation can be entered separately for each axis.

Backup battery

The backup battery ensures that the → user program in the → CPU is stored so that it is safe from power failure and so that specified data areas and bit memory, timers and counters are stored retentively.

Basic axis

Axis whose setpoint or actual value position forms the basis of the calculation of a compensation value.

Basic Coordinate System

Cartesian coordinate system which is mapped by transformation onto the machine coordinate system.

The programmer uses axis names of the basic coordinate system in the → part program. The basic coordinate system exists parallel to the → machine coordinate system if no → transformation is active. The difference lies in the → axis names.
Baud rate
Rate of data transfer (bits/s).

Blank
Workpiece as it is before it is machined.

Block
"Block" is the term given to any files required for creating and processing programs.

Block search
For debugging purposes or following a program abort, the "Block search" function can be used to select any location in the part program at which the program is to be started or resumed.

Booting
Loading the system program after power ON.

C axis
Axis around which the tool spindle describes a controlled rotational and positioning motion.

C spline
The C spline is the most well-known and widely used spline. The transitions at the interpolation points are continuous, both tangentially and in terms of curvature. 3rd order polynomials are used.

Channel
A channel is characterized by the fact that it can process a → part program independently of other channels. A channel exclusively controls the axes and spindles assigned to it. Part program runs of different channels can be coordinated through → synchronization.

Circular interpolation
The → tool moves on a circle between specified points on the contour at a given feedrate, and the workpiece is thereby machined.

CNC
See → NC
Computerized Numerical Control: includes the components → NCK, → PLC, HMI, → COM.
Glossary

CNC
See → NC
Computerized Numerical Control: includes the components → NCK, → PLC, HMI, → COM.

COM
Component of the NC for the implementation and coordination of communication.

Compensation axis
Axis with a setpoint or actual value modified by the compensation value

Compensation table
Table containing interpolation points. It provides the compensation values of the compensation axis for selected positions on the basic axis.

Compensation value
Difference between the axis position measured by the encoder and the desired, programmed axis position.

Continuous-path mode
The objective of continuous-path mode is to avoid substantial deceleration of the → path axes at the part program block boundaries and to change to the next block at as close to the same path velocity as possible.

Contour
Contour of the → workpiece

Contour monitoring
The following error is monitored within a definable tolerance band as a measure of contour accuracy. An unacceptably high following error can cause the drive to become overloaded, for example. In such cases, an alarm is output and the axes are stopped.

Coordinate system
See → Machine coordinate system, → Workpiece coordinate system

CPU
Central processing unit, see → PLC
Curvature

The curvature $k$ of a contour is the inverse of radius $r$ of the nestling circle in a contour point ($k = 1/r$).

Cycles

Protected subprograms for execution of repetitive machining operations on the → workpiece.

Data block

1. Data unit of the → PLC that → HIGHSTEP programs can access.
2. Data unit of the → NC: Data modules contain data definitions for global user data. This data can be initialized directly when it is defined.

Data word

Two-byte data unit within a → data block.

Diagnostics

1. Operating area of the controller.
2. The controller has a self-diagnostics program as well as test functions for servicing purposes: status, alarm, and service displays.

Dimensions specification, metric and inches

Position and pitch values can be programmed in inches in the machining program. Irrespective of the programmable dimensions ($G70/G71$), the controller is set to a basic system.

DRF

Differential Resolver Function: NC function which generates an incremental zero offset in Automatic mode in conjunction with an electronic handwheel.

Drive

The drive is the unit of the CNC that performs the speed and torque control based on the settings of the NC.

Dynamic feedforward control

Inaccuracies in the → contour due to following errors can be practically eliminated using dynamic, acceleration-dependent feedforward control. This results in excellent machining accuracy even at high → path velocities. Feedforward control can be selected and deselected on an axis-specific basis via the → part program.
Editor

The editor makes it possible to create, edit, extend, join, and import programs/texts/program blocks.

Exact stop

When an exact stop statement is programmed, the position specified in a block is approached exactly and, if necessary, very slowly. To reduce the approach time, exact stop limits are defined for rapid traverse and feed.

Exact stop limit

When all path axes reach their exact stop limits, the controller responds as if it had reached its precise destination point. A block advance of the part program occurs.

External zero offset

Zero offset specified by the PLC.

Fast retraction from the contour

When an interrupt occurs, a motion can be initiated via the CNC machining program, enabling the tool to be quickly retracted from the workpiece contour that is currently being machined. The retraction angle and the distance retracted can also be parameterized. An interrupt routine can also be executed following the fast retraction.

Feed override

The programmed velocity is overridden by the current velocity setting made via the machine control panel or from the PLC (0 to 200%). The feedrate can also be corrected by a programmable percentage factor (1 to 200%) in the machining program.

Finished-part contour

Contour of the finished workpiece. See Raw part.

Fixed machine point

Point that is uniquely defined by the machine tool, e.g. machine reference point.

Fixed-point approach

Machine tools can approach fixed points such as a tool change point, loading point, pallet change point, etc. in a defined way. The coordinates of these points are stored in the controller. The controller moves the relevant axes in rapid traverse, whenever possible.
Frame
A frame is an arithmetic rule that transforms one Cartesian coordinate system into another Cartesian coordinate system. A frame contains the following components: → zero offset, → rotation, → scaling, → mirroring.

Functionality
The path-jerk limitation can be activated/deactivated by programming the setting data.
Parameter: Value
• Value range: TRUE, FALSE
Application:
• Part program
• Static synchronized action

Geometry
Description of a → workpiece in the → workpiece coordinate system.

Geometry axis
The geometry axes form the 2 or 3-dimensional → workpiece coordinate system in which, in → part programs, the geometry of the workpiece is programmed.

Ground
Ground is taken as the total of all linked inactive parts of a device which will not become live with a dangerous contact voltage even in the event of a malfunction.

Helical interpolation
The helical interpolation function is ideal for machining internal and external threads using form milling cutters and for milling lubrication grooves.
The helix comprises two motions:
• Circular motion in one plane
• A linear motion perpendicular to this plane

High-level CNC language
The high-level language is used to write NC programs, → synchronized actions, and → cycles. It provides: control structures → user-defined variables, → system variables, → macro programming.
**High-speed digital inputs/outputs**

The digital inputs can be used for example to start fast CNC program routines (interrupt routines). High-speed, program-driven switching functions can be initiated via the digital CNC outputs.

**HIGHSTEP**

Summary of programming options for → PLCs of the AS300/AS400 system.

**HW Config**

SIMATIC S7 tool for the configuration and parameterization of hardware components within an S7 project.

**Identifier**

In accordance with DIN 66025, words are supplemented using identifiers (names) for variables (arithmetic variables, system variables, user variables), subprograms, key words, and words with multiple address letters. These supplements have the same meaning as the words with respect to block format. Identifiers must be unique. It is not permissible to use the same identifier for different objects.

**Inch measuring system**

Measuring system which defines distances in inches and fractions of inches.

**Inclined surface machining**

Drilling and milling operations on workpiece surfaces that do not lie in the coordinate planes of the machine can be performed easily using the function "inclined-surface machining".

**Increment**

Travel path length specification based on number of increments. The number of increments can be stored as → setting data or be selected by means of a suitably labeled key (i.e. 10, 100, 1000, 10000).

**Incremental dimension**

Also incremental dimension: A destination for axis traversal is defined by a distance to be covered and a direction referenced to a point already reached. See → Absolute dimension.

**Intermediate blocks**

Motions with selected → tool offset (G41/G42) may be interrupted by a limited number of intermediate blocks (blocks without axis motions in the offset plane), whereby the tool offset can still be correctly compensated for. The permissible number of intermediate blocks which the controller reads ahead can be set in system parameters.
Interpolator

Logic unit of the → NCK that defines intermediate values for the motions to be carried out in individual axes based on information on the end positions specified in the part program.

Interpolatory compensation

Mechanical deviations of the machine are compensated for by means of interpolatory compensation functions, such as → leadscrew error, sag, angularity, and temperature compensation.

Interrupt routine

Interrupt routines are special → subprograms that can be started by events (external signals) in the machining process. A part program block which is currently being worked through is interrupted and the position of the axes at the point of interruption is automatically saved.

Inverse-time feedrate

The time required for the path of a block to be traversed can also be programmed for the axis motion instead of the feed velocity (G93).

JOG

Control operating mode (setup mode): In JOG mode, the machine can be set up. Individual axes and spindles can be traversed in JOG mode by means of the direction keys. Additional functions in JOG mode include: → Reference point approach, → Repos, and → Preset (set actual value).

Key switch

The key switch on the → machine control panel has four positions that are assigned functions by the operating system of the controller. The key switch has three different colored keys that can be removed in the specified positions.

Keywords

Words with specified notation that have a defined meaning in the programming language for → part programs.

KÜ

Transformation ratio

KV

Servo gain factor, a control variable in a control loop.
Leading axis
The leading axis is the → gantry axis that exists from the point of view of the operator and programmer and, thus, can be influenced like a standard NC axis.

Leadscrew error compensation
Compensation for the mechanical inaccuracies of a leadscrew participating in the feed. The controller uses stored deviation values for the compensation.

Limit speed
Maximum/minimum (spindle) speed: The maximum speed of a spindle can be limited by specifying machine data, the → PLC or → setting data.

Linear axis
In contrast to a rotary axis, a linear axis describes a straight line.

Linear interpolation
The tool travels along a straight line to the destination point while machining the workpiece.

Load memory
The load memory is the same as the → working memory for the CPU 314 of the → PLC.

Look Ahead
The Look Ahead function is used to achieve an optimal machining speed by looking ahead over an assignable number of traversing blocks.

Machine axes
Physically existent axes on the machine tool.

Machine control panel
An operator panel on a machine tool with operating elements such as keys, rotary switches, etc., and simple indicators such as LEDs. It is used to directly influence the machine tool via the PLC.

Machine coordinate system
A coordinate system, which is related to the axes of the machine tool.
Machine zero

Fixed point of the machine tool to which all (derived) measuring systems can be traced back.

Machining channel

A channel structure can be used to shorten idle times by means of parallel motion sequences, e.g. moving a loading gantry simultaneously with machining. Here, a CNC channel must be regarded as a separate CNC control system with decoding, block preparation and interpolation.

Macro techniques

Grouping of a set of statements under a single identifier. The identifier represents the set of consolidated statements in the program.

Main block

A block prefixed by ":" introductory block, containing all the parameters required to start execution of a part program.

Main program

The term "main program" has its origins during the time when part programs were split strictly into main and subprograms. This strict division no longer exists with today's SINUMERIK NC language. In principle, any part program in the channel can be selected and started. It then runs through in program level 0 (main program level). Further part programs or cycles as subprograms can be called up in the main program.

MDA

Control operating mode: Manual Data Automatic. In the MDA mode, individual program blocks or block sequences with no reference to a main program or subprogram can be input and executed immediately afterwards through actuation of the NC start key.

Messages

All messages programmed in the part program and alarms detected by the system are displayed on the operator panel in plain text with date and time and the corresponding symbol for the cancel criterion. Alarms and messages are displayed separately.

Metric measuring system

Standardized system of units: For length, e.g. mm (millimeters), m (meters).

Mirroring

Mirroring reverses the signs of the coordinate values of a contour, with respect to an axis. It is possible to mirror with respect to more than one axis at a time.
Mode
An operating concept on a SINUMERIK controller. The following modes are defined: → Jog, → MDA, → Automatic.

Mode group
Axes and spindles that are technologically related can be combined into one mode group. Axes/spindles of a mode group can be controlled by one or more → channels. The same → mode type is always assigned to the channels of the mode group.

NC
Numerical Control component of the → CNC that executes the → part programs and coordinates the movements of the machine tool.

Network
A network is the connection of multiple S7-300 and other end devices, e.g. a programming device via a → connecting cable. A data exchange takes place over the network between the connected devices.

NRK
Numeric robotic kernel (operating system of → NCK)

NURBS
The motion control and path interpolation that occurs within the controller is performed based on NURBS (Non Uniform Rational B-Splines). This provides a uniform procedure for all internal interpolations.

OEM
The scope for implementing individual solutions (OEM applications) has been provided for machine manufacturers, who wish to create their own user interface or integrate technology-specific functions in the controller.

Offset memory
Data range in the control, in which the tool offset data is stored.

Oriented spindle stop
Stops the workpiece spindle in a specified angular position, e.g. in order to perform additional machining at a particular location.
**Oriented tool retraction**

*RETTOL*: If machining is interrupted (e.g. when a tool breaks), a program command can be used to retract the tool in a user-specified orientation by a defined distance.

**Overall reset**

In the event of an overall reset, the following memories of the → CPU are deleted:

- → Working memory
- Read/write area of → load memory
- → System memory
- → Backup memory

**Override**

Manual or programmable control feature which enables the user to override programmed feedrates or speeds in order to adapt them to a specific workpiece or material.

**Part program**

Series of statements to the NC that act in concert to produce a particular → workpiece. Likewise, this term applies to execution of a particular machining operation on a given → raw part.

**Part program block**

Part of a → part program that is demarcated by a line feed. There are two types: → main blocks and → subblocks.

**Part program management**

Part program management can be organized by → workpieces. The size of the user memory determines the number of programs and the amount of data that can be managed. Each file (programs and data) can be given a name consisting of a maximum of 24 alphanumeric characters.

**Path axis**

Path axes include all machining axes of the → channel that are controlled by the → interpolator in such a way that they start, accelerate, stop, and reach their end point simultaneously.

**Path feedrate**

Path feed affects → path axes. It represents the geometric sum of the feedrates of the → geometry axes involved.
Path velocity

The maximum programmable path velocity depends on the input resolution. For example, with a resolution of 0.1 mm the maximum programmable path velocity is 1000 m/min.

PCIN data transfer program

PCIN is an auxiliary program for sending and receiving CNC user data (e.g. part programs, tool offsets, etc.) via a serial interface. The PCIN program can run in MS-DOS on standard industrial PCs.

Peripheral module

I/O modules represent the link between the CPU and the process.
I/O modules are:
- Digital input/output modules
- Analog input/output modules
- Simulator modules

PLC

Programmable Logic Controller: Programmable logic controller. Component of NC: Programmable control for processing the control logic of the machine tool.

PLC program memory

SINUMERIK 840D sl: The PLC user program, the user data and the basic PLC program are stored together in the PLC user memory.

PLC programming

The PLC is programmed using the STEP 7 software. The STEP 7 programming software is based on the WINDOWS standard operating system and contains the STEP 5 programming functions with innovative enhancements.

Polar coordinates

A coordinate system which defines the position of a point on a plane in terms of its distance from the origin and the angle formed by the radius vector with a defined axis.

Polynomial interpolation

Polynomial interpolation enables a wide variety of curve characteristics to be generated, such as straight line, parabolic, exponential functions (SINUMERIK 840D sl).
Positioning axis

Axis that performs an auxiliary motion on a machine tool (e.g. tool magazine, pallet transport). Positioning axes are axes that do not interpolate with → path axes.

Pre-coincidence

Block change occurs already when the path distance approaches an amount equal to a specifiable delta of the end position.

Program block

Program blocks contain the main program and subprograms of → part programs.

Program level

A part program started in the channel runs as a → main program on program level 0 (main program level). Any part program called up in the main program runs as a → subprogram on a program level 1 ... n of its own.

Programmable frames

Programmable → frames enable dynamic definition of new coordinate system output points while the part program is being executed. A distinction is made between absolute definition using a new frame and additive definition with reference to an existing starting point.

Programmable logic controller

Programmable logic controllers (PLCs) are electronic controllers, the function of which is stored as a program in the control unit. This means that the layout and wiring of the device do not depend on the function of the controller. The programmable logic control has the same structure as a computer; it consists of a CPU (central module) with memory, input/output modules and an internal bus system. The peripherals and the programming language are matched to the requirements of the control technology.

Programmable working area limitation

Limitation of the motion space of the tool to a space defined by programmed limitations.

Programming key

Characters and character strings that have a defined meaning in the programming language for → part programs.

Protection zone

Three-dimensional zone within the → working area into which the tool tip must not pass.
Quadrant error compensation

Contour errors at quadrant transitions, which arise as a result of changing friction conditions on the guideways, can be virtually entirely eliminated with the quadrant error compensation. Parameterization of the quadrant error compensation is performed by means of a circuit test.

R parameters

Arithmetic parameter that can be set or queried by the programmer of the → part program for any purpose in the program.

Rapid traverse

The highest traverse rate of an axis. For example, rapid traverse is used when the tool approaches the → workpiece contour from a resting position or when the tool is retracted from the workpiece contour. The rapid traverse velocity is set on a machine-specific basis using a machine data element.

Reference point

Machine tool position that the measuring system of the → machine axes references.

Rotary axis

Rotary axes apply a workpiece or tool rotation to a defined angular position.

Rotation

Component of a → frame that defines a rotation of the coordinate system around a particular angle.

Rounding axis

Rounding axes rotate a workpiece or tool to an angular position corresponding to an indexing grid. When a grid index is reached, the rounding axis is “in position”.

RS-232-C

Serial interface for data input/output. Machining programs as well as manufacturer and user data can be loaded and saved via this interface.

Safety functions

The controller is equipped with permanently active monitoring functions that detect faults in the → CNC, the → PLC, and the machine in a timely manner so that damage to the workpiece, tool, or machine is largely prevented. In the event of a fault, the machining operation is interrupted and the drives stopped. The cause of the malfunction is logged and output as an alarm. At the same time, the PLC is notified that a CNC alarm has been triggered.
Scaling
Component of a → frame that implements axis-specific scale modifications.

Setting data
Data which communicates the properties of the machine tool to the NC as defined by the system software.

Softkey
A key, whose name appears on an area of the screen. The choice of softkeys displayed is dynamically adapted to the operating situation. The freely assignable function keys (softkeys) are assigned defined functions in the software.

Software limit switch
Software limit switches limit the traversing range of an axis and prevent an abrupt stop of the slide at the hardware limit switch. Two value pairs can be specified for each axis and activated separately by means of the → PLC.

Spline interpolation
With spline interpolation, the controller can generate a smooth curve characteristic from only a few specified interpolation points of a set contour.

Standard cycles
Standard cycles are provided for machining operations which are frequently repeated:
- For the drilling/milling technology
- For turning technology
The available cycles are listed in the "Cycle support" menu in the "Program" operating area. Once the desired machining cycle has been selected, the parameters required for assigning values are displayed in plain text.

Subblock
Block preceded by "N" containing information for a sequence, e.g. positional data.

Subprogram
The term "subprogram" has its origins during the time when part programs were split strictly into →main and subprograms. This strict division no longer exists with today's SINUMERIK NC language. In principle, any part program or any → cycle can be called up as a subprogram within another part program. It then runs through in the next → program level (x+1) (subprogram level (x+1)).
Synchronization

Statements in → part programs for coordination of sequences in different → channels at certain machining points.

Synchronized actions

1. Auxiliary function output
   During workpiece machining, technological functions (→ auxiliary functions) can be output from the CNC program to the PLC. For example, these auxiliary functions are used to control additional equipment for the machine tool, such as quills, grabbers, clamping chucks, etc.

2. Fast auxiliary function output
   For time-critical switching functions, the acknowledgement times for the → auxiliary functions can be minimized and unnecessary hold points in the machining process can be avoided.

Synchronized axes

Synchronized axes take the same time to traverse their path as the geometry axes take for their path.

Synchronized axis

A synchronized axis is the → gantry axis whose set position is continuously derived from the motion of the → leading axis and is, thus, moved synchronously with the leading axis. From the point of view of the programmer and operator, the synchronized axis "does not exist".

Syntax

$SC\_IS\_SD\_MAX\_PATH\_JERK = value$

System memory

The system memory is a memory in the CPU in which the following data is stored:

- Data required by the operating system
- The operands timers, counters, markers

System variable

A variable that exists without any input from the programmer of a → part program. It is defined by a data type and the variable name preceded by the character $. See → User-defined variable.

Tapping without compensating chuck

This function allows threads to be tapped without a compensating chuck. By using the interpolating method of the spindle as a rotary axis and the drilling axis, threads can be cut to a precise final drilling depth, e.g. for blind hole threads (requirement: spindles in axis operation).
Text editor

See → Editor

TOA area

The TOA area includes all tool and magazine data. By default, this area coincides with the → channel area with regard to the access of the data. However, machine data can be used to specify that multiple channels share one → TOA unit so that common tool management data is then available to these channels.

TOA unit

Each → TOA area can have more than one TOA unit. The number of possible TOA units is limited by the maximum number of active → channels. A TOA unit includes exactly one tool data block and one magazine data block. In addition, a TOA unit can also contain a toolholder data block (optional).

Tool

Active part on the machine tool that implements machining (e.g. turning tool, milling tool, drill, LASER beam, etc.).

Tool nose radius compensation

Contour programming assumes that the tool is pointed. Because this is not actually the case in practice, the curvature radius of the tool used must be communicated to the controller which then takes it into account. The curvature center is maintained equidistantly around the contour, offset by the curvature radius.

Tool offset

Consideration of the tool dimensions in calculating the path.

Tool radius compensation

To directly program a desired → workpiece contour, the control must traverse an equistant path to the programmed contour taking into account the radius of the tool that is being used (G41/ G42).

Transformation

Additive or absolute zero offset of an axis.

Travel range

The maximum permissible travel range for linear axes is ± 9 decades. The absolute value depends on the selected input and position control resolution and the unit of measurement (inch or metric).
User interface

The user interface (UI) is the display medium for a CNC in the form of a screen. It features horizontal and vertical softkeys.

User memory

All programs and data, such as part programs, subprograms, comments, tool offsets, and zero offsets/frames, as well as channel and program user data, can be stored in the shared CNC user memory.

User program

User programs for the S7-300 automation systems are created using the programming language STEP 7. The user program has a modular layout and consists of individual blocks.

The basic block types are:

- Code blocks
  These blocks contain the STEP 7 commands.
- Data blocks
  These blocks contain constants and variables for the STEP 7 program.

User-defined variable

Users can declare their own variables for any purpose in the part program or data block (global user data). A definition contains a data type specification and the variable name. See System variable.

Variable definition

A variable definition includes the specification of a data type and a variable name. The variable names can be used to access the value of the variables.

Velocity control

In order to achieve an acceptable traverse rate in the case of very slight motions per block, an anticipatory evaluation over several blocks (Look Ahead) can be specified.

WinSCP

WinSCP is a freely available open source program for Windows for the transfer of files.

Working area

Three-dimensional zone into which the tool tip can be moved on account of the physical design of the machine tool. See Protection zone.
Working area limitation

With the aid of the working area limitation, the traversing range of the axes can be further restricted in addition to the limit switches. One value pair per axis may be used to describe the protected working area.

Working memory

The working memory is a RAM in the CPU that the processor accesses when processing the application program.

Workpiece

Part to be made/machined by the machine tool.

Workpiece contour

Set contour of the workpiece to be created or machined.

Workpiece coordinate system

The workpiece coordinate system has its starting point in the workpiece zero-point. In machining operations programmed in the workpiece coordinate system, the dimensions and directions refer to this system.

Workpiece zero

The workpiece zero is the starting point for the workpiece coordinate system. It is defined in terms of distances to the machine zero.

Zero offset

Specifies a new reference point for a coordinate system through reference to an existing zero point and a frame.

1. Settable
   A configurable number of settable zero offsets are available for each CNC axis. The offsets - which are selected by means of G functions - take effect alternatively.

2. External
   In addition to all the offsets which define the position of the workpiece zero, an external zero offset can be overridden by means of the handwheel (DRF offset) or from the PLC.

3. Programmable
   Zero offsets can be programmed for all path and positioning axes using the TRANS statement.
$A_MONIFACT, 1535
$AA_ACC, 1355
$AA_ATOL, 234
$AA_ETRANS, 769
$AA_FGROUP, 335
$AA_MOTEND, 1356
$AA_S, 1204
$AA_SCPAR, 1357
$AA_VLFCT, 1287
$AA_VMAXB, 1287
$AA_VMAXM, 1287
$AC_ACT_PROG_NET_TIME, 642
$AC_ACTUAL_PARTS, 648
$AC_ASUP, 573
$AC_AUXFU_EXT, 440
$AC_AUXFU_M_EXT, 440
$AC_AUXFU_M_STATE, 441
$AC_AUXFU_M_TICK, 421
$AC_AUXFU_M_VALUE, 441
$AC_AUXFU_PREDEF_INDEX, 426, 440
$AC_AUXFU_SPEC, 438, 441
$AC_AUXFU_STATE, 441
$AC_AUXFU_TYPE, 440
$AC_AUXFU_VALUE, 441
$AC_CONSTCUT_S, 1281
$AC_CUT_INV, 1479
$AC_CUTMOD, 1479
$AC_CUTMOD_ANG, 1473, 1479
$AC_CUTTING_TIME, 644
$AC_CYCLE_TIME, 644
$AC_FGROUP_MASK, 335
$AC_OLD_PROG_NET_TIME, 642
$AC_OLD_PROG_NET_TIME_COUNT, 643
$AC_OPERATING_TIME, 644
$AC_OTOL, 234
$AC_PATHACC, 254, 266
$AC_PATHJERK, 265, 266, 267
$AC_PROG_NET_TIME_TRIGGER, 643
$AC_REQUIRED_PARTS, 648
$AC_SGEAR, 1246, 1266
$AC_SMAXACC, 1287
$AC_SMAXACC_INFO, 1287
$AC_SMAXVELO, 1287
$AC_SMAXVELO_INFO, 1287
$AC_SMINVELO, 1287
$AC_SMINVELO_INFO, 1287
$AC_SPECIAL_PARTS, 648
$AC_SPIND_STATE, 1287
$AC_STOLF, 238
$AC_TOTAL_PARTS, 648
$AN_AUXFU_LIST_ENDINDEX, 431
$AN_POWERON_TIME, 641
$AN_SETUP_TIME, 641
$C_AUX_EXT, 623
$C_AUX_IS_QUICK, 623
$C_AUX_VALUE, 623
$C_D, 624
$C_D_PROG, 624
$C_DL, 624
$C_DL_PROG, 624
$C_DUPLO, 624
$C_DUPLO_PROG, 624
$C_M, 623
$C_M_PROG, 623
$C_ME, 623
$C_MTL, 620, 624
$C_MTL_PROG, 620, 624
$C_T, 623
$C_T_PROG, 623
$C_TCA, 624
$C_TE, 624
$C_THNO, 624
$C_THNO_PROG, 624
$C_TS, 624
$C_TS_PROG, 624
$P_AD, 1479
$P_CHANNO, 553
$P_CONSTCUT_S, 1281
$P_CUT_INV, 1479
$P_CUTMOD, 1479
$P_CUTMOD_ANG, 1473, 1479
$P_FGROUP_MASK, 336
$P_GEAR, 1266
$P_GFRNUM, 731
$P_IFRAME, 729
$P_ISTEST, 472
$P_OTOL, 234
$P_PROG_EVENT, 553
$P_REPINF, 569
$P_SEARCH_S, 424, 484
$P_SEARCH_SDIR, 424, 484
$P_SEARCH_SGEAR, 424, 484
$P_SEARCH_SGEAR, 424, 484, 1246
$P\_SEARCH\_SMODE, 484
$P\_SEARCH\_SPOS, 424, 484
$P\_SEARCH\_SPOSMODE, 424, 485
$P\_SGEAR, 1246
$P\_SIM, 478
$P\_STOLF, 238
$P\_SUB\_AUTOGEAR, 632
$P\_SUB\_AXFCT, 632
$P\_SUB\_CA, 632
$P\_SUB\_GEAR, 632
$P\_SUB\_LA, 632
$P\_SUB\_M19, 633
$P\_SUB\_SPOS, 632
$P\_SUB\_SPOSA, 633
$P\_SUB\_SPOSIT, 633
$P\_SUB\_SPOSMODE, 633
$P\_SUB\_STAT, 622, 624
$P\_TOOLENV, 1519
$P\_TOOLENVN, 1519
$P\_UIFR, 729
$P\_UIFRNUM, 730
$P\_WORKAREA\_CS\_COORD\_SYSTEM, 111
$PA\_ATOL, 234
$PA\_FGREF, 335
$PA\_FGROUP, 336
$SC\_IS\_SD\_MAX\_PATH\_ACCEL, 252
$SC\_IS\_SD\_MAX\_PATH\_JERK, 264
$SC\_SD\_MAX\_PATH\_ACCEL, 252
$SC\_SD\_MAX\_PATH\_JERK, 264
$TC\_DP1, 1469
$TC\_DP10, 1471
$TC\_DP11, 1470
$TC\_DP2, 1470, 1474
$TC\_DP24, 1471
$TC\_DPCE[t,d], 1373
$TC\_DPNT, 1311
$VA\_ABSOLUTE\_ENC\_DELTA\_INIT, 102
$VA\_ENC\_ZERO\_MON\_ERR\_CNT, 99, 102
$VA\_TORQUE\_AT\_LIMIT, 313
$VC\_SGEAR, 1245

ACN, 1273
ACP, 1273
Action blocks, 481
Action single block, 473
Activation
  from machine control panel, hand-held unit, 868
Actual image number of the JobShop interface, 1562
Actual value synchronization, 46
Actual values in workpiece coordinate system, 39
Actual-value acquisition, 337
Actual-value correction, 338
Actual-value processing, 345
Actual-value system
  workpiecerelated, 788
adaptation factor
dynamic path response, 215
Adapter dimension, 1399
Adapting the motor/load ratios, 342
ADDFRAME, 767
AG\_SEND, AG\_RECV, 858
Air temperature alarm, 38
Alarm for preprocessing stop, 1437
ALF, 568, 1322
Angle
  Clearance, 1471
  Holder, 1471
ANY, 1067
ANY in FB, 1069
ANY in FC, 1068
Applied position difference, 370
Assignment tool/toolholder, 1443
ASUB
  Activation, 561
  For user alarms, 574
  Internal, 571
  Priority, 568
  Reorganization, 561
  SERUPRO end, 431
ATRANS, 664
Autonomous singleaxis operations, 523
AUXFUDEL, 432
AUXFUDELG, 433
AUXFUMSEQ, 421
AUXFUSYNC, 432
Auxiliary function
  Associated, 408
Auxiliary function
  Address extension, 400
    -counter, 427
    Definition, 378
    Output behavior, 401
    Predefined, 377

3
3D face milling, 1397

A
ACC, 1275, 1354
Access authorization, 60
Access features, 61
Access security, 60
Type, 400
User-defined, 377
Value, 400
Auxiliary function output, 527
auxiliary functions
  Pre-defined, 384
  User-specific, 405
Axis clamping, 86
Axis configuration, 675
Axis disable, 41, 1335
Axis monitoring functions, 79
  Actual velocity, 94
  Axis/spindlespecific setting data, 174
  Following error, 80
  Speed setpoint, 93
  Supplementary conditions, 159
  Zero speed, 84
Axis/spindle stationary, 49

B
Basic block display
  Activating, 593
  Configure, 592
Basic coordinate system (BCS), 663, 696
Basic display
  Size of display buffer, 593
Basic orientation of tool, example, 1538
Basic orientation of tools, 1485
Basic tool orientation, 1484
Block
  Hide, 589
Block search
  Cascaded, 480
    with calculation at block end point (type 4), 479
    with calculation at the contour (type 2), 479
    with calculation in program test mode, SERUPRO (type 5), 479
    without calculation (type 1), 479
Block search SERUPRO
  Conditions for axis functions, 522
  Control REPOS with NC/PLC interface signals, 501
  Delayed approach of axis with REPOS offset, 499
  Gear stage change, 523
  Initial setting, 525
  Overlaid movements, 524
  Path axes, 500
  Prefer or ignore REPOS, 499
  REPOS acknowledgements, 502
  REPOS offset after an axis interchange, 504
  REPOS offset with synchronous spindle coupling, 505
  Reposition positioning axes, 499
  Set REPOS response, 497
  Setpoint and actual value couplings, 519
  Time sequence, 494
  Updating the REPOS offset within the scope, 504
Block search with calculation
  accumulated spindle functions, 484
Block-related limit (FOC), 312
BLSYNC, 568
BRISK, 247
BRISKA, 247

C
Cancel alarms, 39
Cascaded block search, 487
CFC, 1309
CFCIN, 1309
CFINE, 664
CFTCP, 1309
Channel
  Configuration, 468
  Current, 553
    -display status, 534
  Initial setting, 527
  Properties, 467
    -states, 534
Channel axes, 670
Channel state
  Channel active, 463
  Channel interrupted, 463
  Channel reset, 463
  Channel-specific NCK alarm is active, 38
  Clamping monitoring, 86
  Clearance angle, 1471
  Closed-loop control, 357
  CLRINT, 569
  Coarse offset, 664
  Collision detection, 1430
  COMPCAD, 227
  COMPCURV, 227
  COMPOF, 229, 232
  COMPON, 227
  Computational resolution, 320
  Concurrent axes, 856
  Constant curvature, 1405
  Constant cutting rate, 1312
  Constant cutting speed setting, 1280
  Constant speed, 1314
  Constant tangent, 1405
Continuous-path mode, 184
  Implicit, 187
Contour
  sampling factor, 225
  sampling time, 225
  tolerance, 232
Contour error, 80
Control direction, 345
Control system response
  at the end of the part program, 609
  at the start of the part program, 609
  during run-up, 609
  When resetting:, 609
Controller enable, 45
Controller parameter set switchover, 70
Conversion of basic system, 329
Convex thread, 1326
Counter pulse, 649
CTRANS, 664
Current controller active, 49
Curvature, 224
Cut direction, 1470
Cutting edge
  Position, 1470, 1474
Cutting edge number, 1373
Cutting edge position, 1393
  relevant, 1402
Cutting edges
  Center point, 1473
    -reference point, 1473
    -shape, 1472
Cyclic operation, 851
Cyclic signal exchange, 36

D
D functions, 382, 1367
D number structure
  - flat (without tool management), 1382
D numbers
  Allocation of free ..., 1372
D/DL function replacement, 620
Darken screen, 38
Data channel, faster, 57
Data exchange
  with operator panel, 833
Data interface, 836
DB 31, ...
DBX60.6, 1292
DBX60.7, 1292
DBX83.1, 1215

DB10
DBB4-7, 806
DBX103.0, 37, 1551
DBX103.5, 1552
DBX103.6, 1552
DBX103.7, 1552
DBX104.3, 870
DBX104.4, 870
DBX104.7, 37, 855, 1552, 1553
DBX106.1, 805, 807, 808, 1631
DBX107.7, 331
DBX108.3, 37, 1553
DBX108.5, 37, 1553
DBX108.6, 37, 1553
DBX108.7, 37, 1554
DBX109.0, 38, 1554
DBX109.5, 1555
DBX109.6, 38, 1555
DBX109.7, 38, 1555, 1556
DBX180.2, 1553
DBX56.1, 45, 530, 805, 806, 808, 1629, 1630
DBX56.2, 805, 807, 1630, 1631
DBX56.4, 63, 1551
DBX56.5, 63, 1551
DBX56.6, 63, 1551
DBX56.7, 63, 1551
DBX92.0, 854
DBX92.1, 854
DBX92.2, 854
DBX92.4, 854
DBX92.5, 854
DBX92.6, 854

DB11
DBX0.0, 462, 464, 1601
DBX0.1, 462, 1150, 1152, 1601
DBX0.2, 462, 1150, 1152, 1601
DBX0.4, 467, 1602
DBX0.5, 460, 1602
DBX0.6, 460, 1603
DBX0.7, 460, 533, 807, 1152, 1153, 1154, 1603
DBX07.7, 530
DBX1.0, 1603
DBX1.0 - DBX1.2, 463
DBX1.1, 1604
DBX1.2, 1150, 1152, 1604
DBX1.6, 587, 1604
DBX1.7, 587, 1605
DBX26.4, 465
DBX26.5, 465
DBX4.0, 1605
DBX4.0 - DBX4.2, 462
DBX4.1, 1605
| DBX4.2, 1606 | DBX20.7, 1558, 1562 |
| DBX4.4, 530 | DBX26.1, 1562, 1563 |
| DBX46.4, 465 | DBX26.2, 1563 |
| DBX46.5, 465 | DBX26.3, 1563 |
| DBX5.0, 1606 | DBX26.5, 1563 |
| DBX5.0 - DBX5.2, 463 | DBX26.6, 1564 |
| DBX5.1, 1606 | DBX26.7, 1564 |
| DBX5.2, 1606, 1607 | DBX44.0, 1560 |
| DBX6.0, 463, 465, 1607 | DBX45.0, 1561 |
| DBX6.1, 463, 1607 | DBX45.1, 1561 |
| DBX6.2, 463, 1607 | DBX45.2, 1561 |
| DBX6.3, 461, 805, 808, 1607, 1608 | DBX45.3, 1561 |
| DBX6.4, 465 | DB21 |
| DBX6.5, 465 | DBX378.1, 567 |
| DBX6.7, 461, 1608 | DB21, ... |
| DBX7.0, 465, 1608 | D35.5, 531 |
| DBX7.0 - DBX7.2, 463 | DBB116 - DBB136, 416 |
| DBX7.1, 1608, 1609 | DBB118, 1598 |
| DBX7.2, 1609 | DBB129, 1598 |
| DB1600, 1081 | DBB140 - DBB157, 1599 |
| DB19 | DBB140 - DBB190, 416 |
| DBB10, 1558 | DBB158 - DBB193, 1599 |
| DBB13, 39 | DBB194, 1275 |
| DBB15, 1560 | DBB194 - DBB206, 416, 1599, 1600 |
| DBB16, 39 | DBB2, 1610 |
| DBB17, 39, 40, 1560 | DBB202, 1275 |
| DBB22, 1562 | DBB208 - DBB271, 1621 |
| DBB24, 1562 | DBB317.1, 1622 |
| DBB26, 40 | DBB376, 548, 1626 |
| DBB27, 40 | DBB4, 1336, 1653, 1654, 1655 |
| DBB6, 1558 | DBB5, 1336, 1656, 1657, 1658 |
| DBB7, 1558 | DBB58, 1596 |
| DBB8, 1558 | DBB58 - DBB67, 415 |
| DBX 0.3, 39 | DBB60 - DBB64, 1597 |
| DBX 0.4, 39 | DBB60 - DBB65, 1596 |
| DBX0.0, 38, 1556 | DBB66 - DBB67, 1597 |
| DBX0.1, 38, 39, 1556 | DBB68 - DBB112, 415 |
| DBX0.2, 38, 39, 1557 | DBB68 - DBB97, 1597 |
| DBX0.3, 1557 | DBB68ff., 1275 |
| DBX0.4, 1557 | DBB98 - DBB115, 1598 |
| DBX0.7, 39, 1558 | DBX0.4, 474, 581, 1609 |
| DBX13.5, 1558, 1559 | DBX0.5, 1609 |
| DBX13.6, 1559 | DBX0.6, 476, 1341, 1652 |
| DBX13.7, 1559 | DBX1.0, 1152, 1632 |
| DBX14.0 - DBX14.6, 1559 | DBX1.1, 1587 |
| DBX14.7, 1559 | DBX1.6, 480, 484, 1610 |
| DBX16.6, 1560 | DBX1.7, 472, 530, 1610 |
| DBX16.7, 1560 | DBX10.0 - DBX11.1, 141, 1587, 1588 |
| DBX20.1, 1561 | DBX12.3, 1334, 1660 |
| DBX20.3, 39, 1557, 1561 | DBX16.3, 1334, 1660 |
| DBX20.4, 39, 1557, 1562 | DBX2.0, 477, 531 |
| DBX20.6, 1562 | DBX2.0 - 7, 589 |
Index

Basic Functions

1708
| DBX10.0, 497, 499, 500, 501, 502, 506, 1627 | DBX3.4, 1348 |
| DBX102.5, 114, 116 | DBX3.5, 1348 |
| DBX102.5/6, 1592 | DBX3.6, 1590 |
| DBX102.6, 114, 116 | DBX3.7, 522 |
| DBX12.0, 103 | DBX30.0, 1277 |
| DBX12.0 - DBX12.1, 1590 | DBX30.1, 1277 |
| DBX12.1, 103 | DBX30.2, 1277 |
| DBX12.2, 104 | DBX30.3, 1277 |
| DBX12.2 - DBX12.3, 1590, 1591 | DBX30.4, 1277 |
| DBX12.3, 104 | DBX39.0, 146 |
| DBX12.7, 1159, 1161, 1634 | DBX4.3, 1253, 1286, 1667, 1668 |
| DBX132.0, 1649 | DBX4.4, 1150 |
| DBX132.1, 1649 | DBX4.7, 1150 |
| DBX132.4, 1649 | DBX60.0, 1642 |
| DBX132.5, 1650 | DBX60.2, 98, 1290 |
| DBX138.4, 1651 | DBX60.2 - DBX60.3, 1591 |
| DBX138.5, 1651 | DBX60.3, 98, 1290 |
| DBX16.0 - DBX16.2, 1205, 1236, 1241, 1242, 1243, 1253, 1255, 1259, 1263, 1638 | DBX60.4, 44, 113, 1151, 1152, 1153, 1154, 1166, 1172, 1173, 1176, 1177, 1178, 1290, 1634 |
| DBX16.3, 1201, 1205, 1236, 1241, 1242, 1243, 1253, 1255, 1259, 1263, 1638 | DBX60.4 - DBX60.5, 1248, 1254, 1255, 1258, 1259 |
| DBX16.4, 1231 | DBX60.5, 44, 113, 808, 1151, 1152, 1153, 1154, 1166, 1172, 1173, 1176, 1177, 1178, 1290, 1635 |
| DBX16.4 - DBX16.5, 1231 | DBX60.6, 44, 85, 92, 1208, 1215, 1219, 1220, 1593 |
| DBX16.5, 1231 | DBX60.7, 44, 85, 92, 1215, 1219, 1220, 1262, 1593 |
| DBX16.6, 1231 | DBX61.0, 40, 48, 1578 |
| DBX16.7, 1286, 1639 | DBX61.3, 41, 49, 1177, 1178, 1579 |
| DBX17.0, 1262 | DBX61.4, 48, 49, 1177, 1204, 1205, 1210, 1236, 1284, 1579 |
| DBX17.4 - DBX17.5, 1258, 1639 | DBX61.5, 45, 46, 49, 1203, 1254, 1258, 1580 |
| DBX17.6, 1268, 1639 | DBX61.6, 46, 49, 1580 |
| DBX18.4, 1252, 1253, 1258, 1640 | DBX61.7, 49, 1581 |
| DBX18.5, 1205, 1236, 1252, 1253, 1254, 1258, 1286, 1641 | DBX62.2, 1312, 1670 |
| DBX18.6, 1253 | DBX62.4, 296, 299, 300, 1595 |
| DBX18.6 - DBX18.7, 1642 | DBX62.5, 300, 304, 1596 |
| DBX18.7, 1253 | DBX64.6, 85, 419 |
| DBX2.1, 41, 43, 45, 113, 115, 297, 1177, 1178, 1258, 1573, 1574 | DBX64.7, 85, 419 |
| DBX2.2, 47, 86, 498, 1204, 1212, 1242, 1286, 1574, 1575, 1636 | DBX68.0 - DBX68.3, 1278 |
| DBX2.3, 86, 92, 1589 | DBX69.0, 49, 71 |
| DBX2.4 - DBX2.7, 1166, 1633, 1634 | DBX69.0 - DBX69.2, 1247, 1581 |
| DBX20.1, 1243, 1576 | DBX69.1, 49, 71 |
| DBX21.0 - DBX21.4, 1576, 1586 | DBX69.2, 49, 71 |
| DBX21.0-4, 50, 51 | DBX7.7, 1286 |
| DBX21.5, 50, 1577 | DBX70.0, 502, 503, 504, 1627 |
| DBX21.6, 50, 1577 | DBX70.1, 502, 503, 504, 1627, 1628 |
| DBX21.7, 45, 50, 301, 1577, 1578 | DBX70.2, 502, 503, 1628 |
| DBX28.7, 470 | DBX71.4, 1231, 1635 |
| DBX3.0, 769, 1629 | DBX71.5, 1231, 1635 |
| DBX3.1, 296, 299, 300, 1595 | DBX72.0, 501, 1628 |
| DBX3.2, 1348 | DBX76.0, 49, 1581 |
| DBX3.3, 1348 | }
DBX76.4, 502, 505, 1629
DBX82.0 - DBX82.2, 1236, 1238, 1239, 1241, 1251, 1254, 1261, 1643
DBX82.3, 1236, 1238, 1239, 1251, 1254, 1255, 1258, 1261, 1643
DBX83.0, 1644
DBX83.1, 1204, 1238, 1240, 1285, 1286, 1644, 1645
DBX83.2, 1238, 1240, 1285, 1286, 1645, 1646
DBX83.5, 1204, 1205, 1209, 1210, 1211, 1284, 1646
DBX84.0, 1203, 1236, 1258, 1261, 1646
DBX84.7, 1236, 1238, 1240, 1285, 1286, 1646
DBX84.7, 1203, 1236, 1258, 1261, 1646
DBX85.5, 1220, 1648
DBX9.0, 47, 49, 70
DBX9.0 - DBX9.2, 1575
DBX9.1, 47, 49, 70
DBX9.2, 47, 49, 70
DBX9.3, 48, 1575
DBX9.2, 51, 1582
DBX93.0 - DBX93.2, 1582
DBX93.5, 37, 45, 51, 1553, 1583
DBX93.6, 50, 51, 1583
DBX93.7, 51, 1583
DBX94.0, 51, 1584
DBX94.1, 52, 1584
DBX94.2, 52, 1585
DBX94.3, 52, 1585
DBX94.4, 52, 1585
DBX94.5, 53, 1585, 1586
DBX94.6, 53, 1586
DB31, ... DBX1.1, 297
DB31, ... DBX62.5, 297
DB31, ...
  DBX9.3, 1357
DB4500, 1082
DB9000 - DB9063, 1082
DC, 1273
Debug mode for SBL2, 582
Decoding single block, 473
Default passwords, 63
Delete distance-to-go, 40
DELTOOLENV, 1518
Description of a rotation, 1454
DIACYCOFA, 818
DIAM90, 817
DIAM90/DIAM90A[AX], 813
DIAM90A, 818
DIAMCHAN, 818
DIAMCHANA, 818
DIAMCYCOF, 817
DIAMCYCOF/DIACYCOFA[AX], 813
Diameter programming
  Axis-specific, 818
  Channel-specific, 817
DIAMOF, 817
DIAMOFA, 818
DIAMON, 817
DIAMON/DIAMONA[AX], 812
DIAMONA, 818
DILF, 1322
Direct keys
  Address, 870
  Alarms, 870
  OPs at Ethernet bus, 870
Direction vector, 1485
DISABLE, 569
DISC, 1425
Display block, structure (DIN), 595
Display resolution, 320
DISPLOF, 584
DISPLOP, 584
DITE, 1318
DITS, 1318
DL functions, 382
DRIVE, 280
Drive ready, 37
Drive test travel enable, 40
Drive test travel request, 48
DRIVEA, 280
Drives in cyclic operation, 37
Dry run feedrate, 475, 1340
Dynamic response
  adaptation, 213
Dynamic response adaptation, 359
DYNFINSIH, 222
DYNNORM, 222
DYNPOS, 222
DYNROUGH, 222
DYNSEMIFIN, 222
E
EES, 603
Emergency stop
  Acknowledgement, 807
  Interface, 805
  Sequence, 805
Emergency stop control elements, 804
ENABLE, 569
Encoder directly at the tool, 344
Encoder monitoring functions, 97
Encoder frequency, 97
End-of-motion criterion
   For single axes, 1356
Ethernet
   -connection, 868
Evaluation of individual wear components, 1514
Event-controlled program sequences, 545
Exact stop, 180
   Implicit, 185
Exact-stop criteria, 181, 182
Execute external subprogram, 599
External program memory, 599
External work offset, 698

F
F functions, 383
FA, 1275, 1333
FA functions, 383
FB, 1352
FB1 RUN_UP (basic program, startup section), 914
FB10 Safety relay, 979
FB11 Brake test, 981
FB2 GET (Read NC Variable), 922
FB29 Signal recorder and data trigger diagnostics, 987
FB3 PUT (write NC variables), 930
FB4 PI_SERV (PI services), 937
   Available PI services, 941
FB5 GETGUD (read GUD variable), 965
FB7 PI_SERV2 (PI services), 973
FB9 MzuN (operator panel switchover), 974
FC10 AL_MSG, 1010
FC1005 AG_SEND, 1061
FC1006 AG_RECV, 1062
FC12 AUXFU, 1012
FC13 BHGDisp, 1013
FC17 Star-delta, 1017
FC18 SpinCtrl, 1020
FC19 MCP_IFM, 1030
FC2 GP_HP (Basic program, cyclic section), 990
FC21 Transfer, 1037
FC22 TM_DIR, 1044
FC24 MCP_IFM2, 1046
FC25 MCP_IFT, 1049
FC3 GP_PRAL (Basic program, diagnostics), 994
FC3 GP_PRAL (Basic program, interrupt driven section), 992
FC6 TM_TRANS2, 996
FC7 TM_REV, 997
FC8 TM_TRANS, 1000
FC9 ASUB, 1007
FCUB, 1349
Feedrate
   for chamfer/rounding, 1350
   Inverse-time (G93), 1310
   Linear (G94), 1311
   -override, 1335
   Path feedrate F, 1308
   Revolutionall (G95), 1311
   Tooth, 1311
   -types, 1307
   types (G93, G94, G95), 1310
Feedrate disable
   Channel-specific, 1334
Feedrate override
   Programmable, 1339
Feedrates
   Dry run feedrate, 1340
   Feedrate disable, 1334
   Feedrate/spindle stop, 1334
FGROUP, 671, 674
FIFO Buffer, 599
Fine interpolation, 358
Fine offset, 664
Firmware, 1079
Fixed feedrates, 1347
FL, 1309
Flat D number structure, 1382
FLIN, 1349
FNORM, 1349
FOC, 294
FOCOF, 294
FOCON, 294
Follow up active, 49
FPO, 1349
FPRAOF, 1274
FPRAON, 1274
FRAME, 670
Frame change, 1445
Frame rotations, 666
   in the direction of the tool, 775, 776
   with solid angles, 773
FRC, 1350
FRCM, 1350
Free-form surface
   mode, 178
Free-form surfaces, 225
   mode, 224
Function interface, 837
Function overview of inch/metric switchover
  Data backup, 332
  Synchronized actions, 327
Function selection (via operator panel front or PLC), 588
FXS, 294
FXS-REPOS, 305
FXST, 294
FXSW, 294
FZ, 1311

G
G groups, 527
G0 tolerance factor, 235
G25, 108, 1274
G26, 108, 1274
G33, 1317
G331, 1331
G332, 1331
G335, 1326
G336, 1326
G40, 1403
G41, 1403
G42, 1403
G450/G451, 1424
G451, 1426
G460, 1439
G461, 1365, 1440
G462, 1365, 1441
G58, 664
G59, 664
G60, 180
G601, 182
G602, 182
G603, 182
G63, 1332
G64, 187
G642, 192
G643, 192
G644, 195
G645, 199
G74, 1154
G9, 180
G91 extension, 1481
  Zero point offset, 1483
G93, 1310
G94, 1311
G95, 1311
G96, 1312
G961, 1312
G97, 1314

G971, 1314
Gear stage
  in M70, 1264
  Manual entry, 1241
  Specification by PLC, 1240
Gear stage change
  Automatic, 1293
Gear stages, 1232
Geometry axes, 670, 696
GETTCOR, 1521
  Compatibility, 1526
  Orientable toolholder, 1524
GETTENV, 1520
GOTOS, 540
Grouping together auxiliary functions, 400

H
H functions, 380
Hardware interrupt, 853
Hardware limit switch, 103
Helical interpolation, 674
Helix interpolation, 674
High-speed data channel, 57
HMI - CPU at OPI ready, 37
HMI monitor, 893
Holder angle, 1471

I
IC, 1273
ID check, 649
Implicit continuous-path mode, 187
Implicit exact stop, 185
Implicit preprocessing stop, 517
Incrementally programmed compensation values, 1481
Input resolution, 320
Interface
  MCP/PLC and HHU/PLC, 835
  PLC/HMI, 835, 842
  PLC/HMI messages, 843
  PLC/MCP, 847
  PLC/NCK, 836
Interface signals
  Acknowledge emergency stop, 1630, 1631
  Acknowledge fixed stop reached, 1594
  Action block active, 1613, 1614
  Activate associated M1, 1596
  Activate channel-specific protection zone, 1587, 1588
Activate dry run feed, 1652
Activate fixed feedrate 1 - 4 for machine axes, 1667
Activate fixed feedrate 1 - 4 for path/geometry axes, 1662
Activate M01, 1609
Activate machine-specific protection zone, 1587
Activate program test, 1610
Activate referencing, 1632
Activate single block, 1609
Activate skip block, 1610
Activate travel to fixed stop, 1595
Active, 1563
Active G function of groups 1 to 60, 1621
Active JOG mode, 1607
Active machine function REF, 1609
Active machine function TEACH IN, 1608
Active mode AUTOMATIC, 1607
Active mode MDA, 1607
Active or passive file system, 1559, 1560
Active REPOS machine function, 1608, 1609
Active spindle control mode, 1648
Active spindle mode oscillation mode, 1647
Active spindle positioning mode, 1647
Actual direction of rotation clockwise, 1646, 1647
Actual gear stage A to C, 1637
Actual values in WCS, 1558
Air temperature alarm, 1555
All axes stationary, 1592
All axes that have to be referenced are referenced, 1633
All channels in the reset state, 1608
Analog measured value of the clamping system, 1651
Analog spindle 1, utilization in percent, 1558
Analog spindle 2, utilization in percent, 1558
Approach block active, 1614
Associated M01/M00 active, 1600
ASUB is active, 1626
ASUB is stopped, 1622, 1623
AT box ready, 1552
AUTOMATIC mode, 1601
Axis/spindle disable, 1567, 1568, 1569
Axis/spindle stationary, 1579
Block search active, 1616
Block search via program test is active, 1623
Cancel alarm deleted, 1561
Change gear, 1643
Channel is ready, 1620
Channel number of the machine control panel to HMI, 1558
Channel status active, 1619
Channel status interrupted, 1619
Channel status reset, 1620
Channel-specific NCK alarm is active, 1565
Channel-specific protection zone preactivated, 1588
Channel-specific protection zone violated, 1589
Clamping in progress, 1589
Clear cancel alarms, 1557
Clear recall alarms, 1557
Control program branching, 1627
Controller enable, 1573, 1574
Controller parameter set switchover (request), 1575
Current controller active, 1581
D function 1, 1598
Darken screen, 1556
Delete distance-to-go (axis-specific) / spindle reset, 1574, 1575
Delete distance-to-go (channel-specific), 1565
Delete S value, 1639
Disable parameter set switchover commands from NC, 1575
Displayed channel number from HMI, 1562
Drive ready, 1583
Drive test travel enable, 1566
Drive test travel request, 1578
Drives in cyclic operation, 1553
Drives ready, 1553
Dry run feedrate selected, 1661
Emergency stop, 1629, 1630
Emergency stop active, 1631
Enable protection zones, 1587
Enable travel to fixed stop, 1595
Encoder limit frequency exceeded, 1591
Error, 1563
Extended address F functions 1 to 6, 1599
Extended address H functions 1 to 3, 1599
Extended address M functions 1 to 5, 1597
Extended address S functions 1 to 3, 1598
External zero offset, 1629
F auxiliary function for positioning axis, 1600
F function for positioning axis, 1670
FC9 Out\ Active, 1561
FC9 Out\ Done, 1561
FC9 Out\ Error, 1561
FC9 Out\ StartError, 1561
Feed stop (Geometry axis 1 to 3), 1660
Feed stop / spindle stop (axis-specific), 1667, 1668
Feedrate disable, 1659
Feedrate override, 1653, 1654, 1655
Feedrate override (axis-specific), 1663, 1664, 1665
Feedrate override active, 1660
Feedrate override selected for rapid traverse, 1661
Fixed stop reached, 1596
Follow up active, 1579
Follow-up mode, 1569, 1570
Gear is changed over, 1638
Hardware limit switches plus and minus, 1590
HMI battery alarm, 1552
HMI CPU Ready, 1553
HMI temperature limit, 1552
Interrupt processing active, 1620
Invert M3/M4, 1639
JOG mode, 1601
Key disable, 1557
Key-operated switch position, 1551
Last action block active, 1615
Load, 1564
Load part program, 1559
Lubrication pulse, 1581
M auxiliary function for spindle, 1600
M fct. 1-5 not decoded, 1597
M function for spindle, 1648
M(d) less than M(dx), 1585
M, S, T, D, H, F functions Additional info "Quick" (quick acknowledgment), 1597
M, S, T, D, H, F functions Modification, 1596
M00/M01 active, 1614, 1615
M02/M30 active, 1616, 1617
Machine function REF, 1604
Machine function REPOS, 1604
Machine function TEACH IN, 1603
Machine-related protection zone preactivated, 1588
Machine-related protection zone violated, 1589
MDA mode, 1601
Mode change disable, 1560, 1602
Mode group ready, 1607, 1608
Mode group reset, 1603
Mode group stop, 1602
Mode group stop axes plus spindles, 1603
Motor being selected, 1577
Motor/drive data set: Display interface, 1582
Motor/drive data set: Formatting interface, 1586
Motor/drive data set: Request interface, 1576
n(act) equals n(set), 1586
n(act) less than n(min), 1585
n(act) less than n(x), 1585, 1586
NC Ready, 1554
NC Start, 1611
NC Start disable, 1611
NC Stop, 1612
NC Stop at block limit, 1612
NC Stop axes plus spindles, 1612
NCK alarm is active, 1554
NCK alarm with processing stop present, 1566
NCK battery alarm, 1555, 1556
NCK CPU Ready, 1552, 1553
NCU heat sink temperature alarm, 1555
OK, 1562, 1563
Oscillation controlled by the PLC, 1640
Oscillation enable, 1641
Override active, 1666
Part program selection, 1559
Path axis, 1629
PLC action completed, 1610
PLC hard keys, 1558
PLC index, 1559, 1560
PLC line offset, 1560
POS_RESTORED 1, 1635
POS_RESTORED2, 1635
Position controller active, 1580
Position measuring system 1/2 activated, 1592
Position measuring systems 1 and 2, 1570, 1571, 1572
Position reached with exact stop coarse, 1593
Position reached with exact stop fine, 1593
ProgEventDisplay, 1626
Program level abort, 1611
Program status aborted, 1619
Program status interrupted, 1619
Program status running, 1617, 1618
Program status stopped, 1618
Program status wait, 1618
Program test active, 1617
Pulse enable, 1577, 1578
Pulses enabled, 1583
Ramp-function generator disable, 1576
Ramp-function generator disable active, 1582
Rapid traverse override, 1656, 1657, 1658
Rapid traverse override active, 1659
Read-in disable, 1611
Read-in enable is ignored, 1620, 1621
Recall alarm deleted, 1562
Reference point approach delay, 1634
Reference point value 1 to 4, 1633, 1634
Referenced/synchronized 1, 1634
Referenced/synchronized 2, 1635
Referencing active, 1632
Remote diagnosis active, 1551
Repos DEFERAL Chan, 1626
REPOS Delay Ackn, 1628
REPOS offset, 1627
REPOS offset valid, 1627, 1628
Repos Path Mode Ackn0-2, 1624, 1625
REPOSDELAY, 1627, 1628
REPOSMODEEDGEACKN, 1624
Reset, 1613
Re-synchronizing spindle when positioning 1 and 2, 1639
Resynchronizing spindles 1 and 2, 1638
Revolutional feed rate active, 1670
Rigid tapping active, 1647
Run-up completed, 1585
S auxiliary function for spindle, 1600
S function for spindle, 1649
Screen bright, 1556
Screen is dark, 1561
Second software limit switch plus or minus, 1590, 1591
Selected JOG mode, 1606
Selected machine function REF, 1606, 1607
Selected mode AUTOMATIC, 1605
Selected mode MDA, 1605
Selected mode TEACH IN, 1606
Selection, 1564
Sensor for fixed stop, 1594
Sensor S1 available (clamped state), 1649
Sensor S4 available (piston position), 1649
Sensor S4, piston end position, 1651
Sensor S5 available (angular position of the motor shaft), 1650
Sensor S5, angular position of the motor shaft, 1651
Sensors available, 1649
Setpoint direction of rotation, counter-clockwise/ setpoint direction of rotation, clockwise, 1642
Setpoint gear stage A to C, 1643
Setpoint speed increased, 1645, 1646
Setpoint speed limited, 1644, 1645
Simulation active, 1562
Single block type A, 1605
Single block type B, 1604
Speed controller active, 1580
Speed controller integrator disable, 1577
Speed controller integrator disabled, 1583
Speed limit exceeded, 1644
Spindle in position, 1648
Spindle in setpoint range, 1646
Spindle override, 1668, 1669
Spindle reset/Delete distance to go, 1636
Spindle/no axis, 1642
State value is generated, speed limitation p5043 is active, 1650
Status of the clamping system, 1650
Still ASUB is active, 1626
Stop at the end of block with SBL is suppressed, 1621
Switch over MCS/WCS, 1562
Synchronized Actions (S5, H2), Signals for dynamic M functions: M0 - M99, 1599, 1600
T function 1, 1598
Transformation active, 1617
Unload part program, 1558, 1559
Unloading, 1563
Velocity/spindle speed limitation, 1590
Workpiece setpoint reached, 1622
Intermediate gear, 343, 1249
Interpolator end, 181
Interpolatory axis grouping, 46
Interrupt
-lock, 569
routine, 559
signal, 559
Interrupt routine
End, 561
Inverse-time feedrate (G93), 1310
IR, 1326

J
Jerk
-increase, velocity-dependent jerk, 268
Jerk limitation, 197
Jogging
during interruption of a JOG ASUB, 565
in the mode type AUTOMATIC, 464
JR, 1326

K
Key disable, 39, 884
Key switch, 63
Kinematic transformation, 696
Kinematic type, 1446
KONT, 1404
KONTC, 1404
KONTT, 1404
KR, 1326

L
LENTOAX, 1531
LFOF, 1322
LFON, 1322
Limit velocity, for path axes, 1309
Limited toolholder orientation, 1445
Limit-switch monitoring, 103
LIMS, 1274
Linear feedrate (G94), 1311
Linear signal distortions, 80
Loader axes, 671
Lockable data areas, 64
LookAhead, 200
Selection and deselection, 202
Lubrication pulse, 49

M
M decoding acc. to list, 860
M function replacement, 618
M1, 443
M17, 443
M19, 1207, 1273
M2, 443
M3, 1273
M30, 443
M4, 1273
M40, 1275, 1293
M41, 1275
M42, 1275
M43, 1275
M44, 1275
M45, 1275
M5, 1273
M70, 1273
Machine axes, 669
Machine coordinate system (MCS), 39, 663, 685
Machine kinematics, 1446
Machine tool axes, 671
Machine with rotary tool, 1450
Machine with rotary workpiece, 1451
Machine zero, 1149
Machine zero M, 678
Machines with extended kinematics, 1452
Machining in direction of tool orientation, 1483
Main axes, 672
Main run, 467
-axes, 1355
Manual switchover of the basic system
   General, 329
Input resolution and computational resolution, 332
JOG and handwheel factor, 332
Reference point, 331
System data, 330
Tool data, 331
Master-slave changeover with G96, G961, 1316
MCP identification, 871
MCS, 39
Md < Mdx, 52
MD10000, 662
MD10010, 458
MD10050, 1189
MD10070, 1189
MD10125, 608
MD10131, 53
MD10192, 1250
MD10200, 318, 319, 321
MD10210, 318, 319, 321, 353, 355
MD10220, 323
MD10230, 323
MD10240, 325, 326, 330, 331, 332, 333, 1281, 1358
MD10260, 329, 332, 333
MD10270, 331
MD10290, 331
MD10292, 331
MD10366, 1084
MD10368, 1084
MD10600, 708, 775
MD10602, 740, 741, 743, 751, 755
MD10610, 666, 737
MD10612, 737
MD10615, 783
MD10618, 143
MD10651, 1326
MD10680, 226
MD10682, 226
MD10702, 475, 482, 567, 581
MD10704, 1341
MD10707, 494
MD10708, 494
MD10710, 110, 1315
MD10712, 223
MD10713, 417
MD10714, 396, 619
MD10715, 618
MD10716, 618
MD10719, 621
MD10735, 464
MD10804, 619
MD10806, 620
MD10814, 620
Index

MD1103, 307
MD1104, 307
MD1105, 307
MD11100, 406
MD11110, 412
MD11220, 333
MD11300, 1150
MD11346, 1381
MD11410, 1236, 1318, 1431
MD11411, 471
MD11450, 423, 484, 485, 487, 496, 521
MD11470, 497, 498, 499, 523
MD11550, 1332
MD11600, 564
MD11602, 563, 566
MD11604, 521, 563, 566
MD11610, 571
MD11620, 549
MD11625, 608
MD11626, 608
MD12000, 1337
MD12010, 1338
MD12020, 1336
MD12030, 203, 204, 1336
MD12040, 1336
MD12050, 1336
MD12060, 1338
MD12070, 1339
MD12080, 1339
MD12082, 1337
MD12090, 1332
MD12100, 203, 204, 1339
MD12200, 1348
MD12202, 1348
MD12204, 1348
MD1230/1231, 307
MD14504, 864
MD14506, 864
MD14508, 864
MD15700, 630
MD15702, 630
MD17200, 593
MD18080, 1498
MD18088, 1447, 1461, 1465
MD18100, 1368, 1383
MD18102, 1375, 1382, 1383
MD18104, 1517
MD18105, 1373, 1374, 1375, 1376
MD18106, 1373, 1374
MD18108, 1502
MD18112, 1503, 1516
MD18114, 1485
MD18116, 1518, 1519, 1520, 1521, 1527
MD18150, 57
MD18190, 132, 144
MD18360, 229, 599
MD18382, 600
MD18600, 706
MD18601, 782
MD18602, 729, 731, 732, 734, 782
MD18960, 247, 262
MD20000, 468
MD20050, 613, 662, 761
MD20060, 662
MD20070, 662
MD20080, 662, 674
MD20090, 442, 1272, 1386
MD20092, 1270
MD20094, 395, 620
MD20095, 395, 620
MD20096, 1370, 1371, 1372
MD20100, 814, 1314, 1316, 1524
MD20105, 521, 565, 570
MD20106, 486, 550, 581
MD20107, 486, 551
MD20108, 549
MD20109, 550
MD20110, 329, 563, 783, 784, 785, 791, 816, 1366, 1385, 1386, 1467, 1482, 1507
MD20112, 495, 563, 613, 786, 787, 791, 816, 1385, 1467
MD20115, 521, 565, 570
MD20116, 566
MD20117, 567, 581
MD20118, 613
MD20120, 613, 1385, 1466
MD20121, 613, 1366, 1385
MD20124, 442
MD20125, 1476
MD20126, 1466, 1476
MD20127, 1476
MD20130, 613, 1385
MD20140, 613
MD20144, 615
MD20150, 110, 238, 247, 262, 280, 281, 325, 330, 468, 527, 613, 721, 785, 786, 816, 1358
MD20152, 613, 785, 816
MD20170, 228
MD20172, 228, 1349
MD20180, 1458
MD20184, 778, 779, 1460
MD20188, 1454
MD20190, 1454
MD20191, 567
<table>
<thead>
<tr>
<th>Page Numbers</th>
<th>Index Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD20192, 551</td>
<td>MD21110, 776</td>
</tr>
<tr>
<td>MD20193, 552</td>
<td>MD21158, 283</td>
</tr>
<tr>
<td>MD20194, 563, 575</td>
<td>MD21159, 283</td>
</tr>
<tr>
<td>MD20200, 1351</td>
<td>MD21166, 282</td>
</tr>
<tr>
<td>MD20201, 1351</td>
<td>MD21168, 283</td>
</tr>
<tr>
<td>MD20202, 1420</td>
<td>MD21202, 567</td>
</tr>
<tr>
<td>MD20204, 1413</td>
<td>MD21220, 1343</td>
</tr>
<tr>
<td>MD20210, 1426</td>
<td>MD21230, 1343</td>
</tr>
<tr>
<td>MD20220, 1425</td>
<td>MD21330, 613</td>
</tr>
<tr>
<td>MD20230, 1427</td>
<td>MD22000, 406, 1207</td>
</tr>
<tr>
<td>MD20240, 1429, 1430</td>
<td>MD22010, 407, 1207</td>
</tr>
<tr>
<td>MD20250, 1404</td>
<td>MD22020, 407, 1207</td>
</tr>
<tr>
<td>MD20252, 1436</td>
<td>MD22030, 406, 1207</td>
</tr>
<tr>
<td>MD20256, 1438</td>
<td>MD22035, 408</td>
</tr>
<tr>
<td>MD20270, 382, 1367, 1378, 1383, 1385, 1507, 1508</td>
<td>MD22040, 399, 406</td>
</tr>
<tr>
<td>MD20272, 382, 1502, 1507, 1508</td>
<td>MD22050, 400</td>
</tr>
<tr>
<td>MD20310, 508</td>
<td>MD22060, 400</td>
</tr>
<tr>
<td>MD20360, 815, 1381, 1516, 1524, 1530, 1546</td>
<td>MD22070, 401</td>
</tr>
<tr>
<td>MD20390, 1494, 1496, 1497</td>
<td>MD22080, 401, 631</td>
</tr>
<tr>
<td>MD20392, 1495</td>
<td>MD22100, 416</td>
</tr>
<tr>
<td>MD20400, 203</td>
<td>MD22110, 381, 384</td>
</tr>
<tr>
<td>MD20430, 204</td>
<td>MD22200, 410</td>
</tr>
<tr>
<td>MD20440, 204</td>
<td>MD22210, 380, 410</td>
</tr>
<tr>
<td>MD20443, 207</td>
<td>MD22220, 381, 410</td>
</tr>
<tr>
<td>MD20450, 205</td>
<td>MD22230, 381, 410</td>
</tr>
<tr>
<td>MD20460, 211, 212</td>
<td>MD22240, 383, 410, 1334</td>
</tr>
<tr>
<td>MD20462, 211</td>
<td>MD22250, 382, 410</td>
</tr>
<tr>
<td>MD20465, 215</td>
<td>MD22252, 383, 410</td>
</tr>
<tr>
<td>MD20480, 193, 195, 232</td>
<td>MD22254, 395, 408, 620</td>
</tr>
<tr>
<td>MD20482, 228, 232, 236</td>
<td>MD22256, 395, 408, 620</td>
</tr>
<tr>
<td>MD20484, 228</td>
<td>MD22410, 1334, 1340, 1358</td>
</tr>
<tr>
<td>MD20485, 228</td>
<td>MD22510, 527, 1065</td>
</tr>
<tr>
<td>MD20486, 228</td>
<td>MD22530, 438, 1467</td>
</tr>
<tr>
<td>MD20487, 228</td>
<td>MD22532, 438</td>
</tr>
<tr>
<td>MD20488, 230</td>
<td>MD22534, 438</td>
</tr>
<tr>
<td>MD20490, 187, 229</td>
<td>MD22550, 621, 1366, 1378, 1379, 1383, 1385</td>
</tr>
<tr>
<td>MD20500, 248, 249</td>
<td>MD22560, 395, 442, 620, 1366, 1385, 1386</td>
</tr>
<tr>
<td>MD20550, 183</td>
<td>MD22562, 1380, 1381</td>
</tr>
<tr>
<td>MD20552, 183</td>
<td>MD22600, 506</td>
</tr>
<tr>
<td>MD20560, 236</td>
<td>MD22620, 525</td>
</tr>
<tr>
<td>MD20600, 261</td>
<td>MD24004, 783</td>
</tr>
<tr>
<td>MD20602, 257, 258</td>
<td>MD24006, 699, 721, 784</td>
</tr>
<tr>
<td>MD20606, 225</td>
<td>MD24007, 786</td>
</tr>
<tr>
<td>MD20610, 251</td>
<td>MD24008, 783</td>
</tr>
<tr>
<td>MD20624, 816</td>
<td>MD24010, 736</td>
</tr>
<tr>
<td>MD20700, 1184</td>
<td>MD24020, 727</td>
</tr>
<tr>
<td>MD20750, 1314</td>
<td>MD24040, 760, 761</td>
</tr>
<tr>
<td>MD20800, 379, 443, 444</td>
<td>MD24100, 1496</td>
</tr>
<tr>
<td>MD20850, 1207</td>
<td>MD24110, 1496</td>
</tr>
<tr>
<td>MD21015, 55</td>
<td>MD24120, 1496</td>
</tr>
<tr>
<td>MD21016, 56</td>
<td>MD24550, 1525</td>
</tr>
<tr>
<td>MD21020, 107</td>
<td>MD24558, 1525</td>
</tr>
</tbody>
</table>
Index

Basic Functions

Function Manual, 01/2015, 6FC5397-0BP40-5BA2

MD24560, 1525
MD24570, 1496
MD24572, 1497
MD24650, 1525
MD24658, 1525
MD24660, 1525
MD24805, 750
MD24855, 750, 751
MD24905, 744
MD24955, 744
MD25574, 1497
MD26008, 396, 620
MD26012, 620
MD27100, 816
MD27800, 469
MD27850, 645
MD27860, 539, 645
MD27880, 540, 648
MD27882, 649
MD28060, 229, 590, 593
MD28070, 229, 230
MD28082, 721, 738, 769, 770, 779, 781, 1460
MD28085, 1368, 1461, 1465
MD28150, 59
MD28200, 132, 144
MD28210, 144
MD28400, 593
MD28520, 229
MD28530, 57, 194, 229
MD28533, 207
MD28540, 229
MD28560, 787
MD28600, 111
MD28610, 225
MD30100, 339
MD30110, 339
MD30120, 339
MD30130, 338, 339
MD30200, 337, 347
MD30210, 339, 348
MD30220, 340, 348
MD30230, 340, 348
MD30240, 338, 340, 348
MD30242, 338, 341, 348, 1175
MD30244, 348
MD30250, 348, 1180
MD30260, 348
MD30270, 348
MD30300, 347, 349, 352, 355, 1358
MD30310, 104, 347
MD30320, 347
MD30330, 347, 1173

MD30340, 347, 1174
MD30350, 338
MD30455, 1174, 1274
MD30460, 814, 818
MD30550, 1151
MD31000, 349, 352, 355
MD31010, 349
MD31020, 349, 353, 355, 1189
MD31025, 353
MD31030, 159, 347, 349, 353
MD31040, 349, 353, 355, 370, 1217, 1258
MD31044, 344, 349, 1249
MD31046, 116
MD31050, 71, 159, 343, 349, 350, 353, 355, 1248, 1254
MD31060, 71, 159, 343, 349, 350, 353, 355, 1248
MD31064, 343, 347, 1249
MD31066, 343, 347, 1249
MD31070, 159, 349, 353, 355
MD31075, 349, 353, 355
MD31090, 332
MD31122, 1230
MD31123, 1230
MD32000, 81, 279, 317, 347, 1290, 1309
MD32040, 1312
MD32050, 1312
MD32060, 523, 1354
MD32074, 783
MD32100, 345
MD32220, 372
MD32220, 372
MD32250, 159, 346
MD32260, 159
MD32300, 81, 246, 255, 256, 257, 258, 279, 1354
MD32301, 282
MD32310, 187, 229, 256, 257
MD32320, 1355
MD32400, 272, 366
MD32402, 273, 366
MD32410, 273, 366
MD32420, 248, 262, 263
MD32430, 262
MD32431, 261, 267
MD32432, 268
MD32433, 256
MD32434, 255, 256, 267
MD32435, 267
MD32436, 282
MD32439, 269
MD32440, 211, 215
MD32610, 81
Index

Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2

MD32620, 363
MD32630, 363
MD32640, 365
MD32711, 331
MD32750, 1494, 1496, 1497
MD32800, 71, 81
MD32810, 71, 81, 159, 363
MD32890, 368, 369
MD32900, 359
MD32910, 71, 359, 364
MD32950, 370
MD33050, 49
MD33100, 193, 226, 228, 232
MD33120, 199, 232
MD34000, 1163, 1173
MD34010, 1150, 1159, 1161, 1162, 1171, 1182
MD34020, 1159, 1162
MD34040, 1161, 1162, 1163, 1171, 1230
MD34050, 1161
MD34060, 1164, 1172, 1173, 1218
MD34070, 1165, 1172
MD34080, 345, 1165
MD34090, 345, 348, 1165, 1168, 1169, 1181, 1183, 1184
MD34092, 1163
MD34093, 1164
MD34100, 1166, 1172, 1182, 1184
MD34102, 1174, 1175
MD34104, 1177
MD34110, 1152
MD34200, 1169, 1175, 1177, 1182, 1185, 1230
MD34210, 1179, 1180, 1181, 1182, 1183, 1184
MD34300, 1171
MD34320, 348, 349, 1168
MD34330, 1186
MD34990, 1309
MD35000, 662, 716, 1269, 1275
MD35010, 1233, 1237, 1251, 1254, 1258, 1263
MD35012, 1232, 1258, 1261
MD35014, 1264
MD35020, 1227
MD35030, 1227
MD35035, 1208, 1265, 1267
MD35040, 1204
MD35090, 1232
MD35092, 1237
MD3510, 1262
MD35100, 317, 1288
MD35110, 1232, 1293
MD35112, 1237
MD35120, 1232, 1293
MD35122, 1237
MD35130, 71, 1232, 1238, 1240, 1266
MD35135, 1232
MD35140, 1232, 1238, 1240, 1285, 1315, 1317
MD35150, 1161, 1209, 1211, 1285, 1289
MD35200, 1203, 1211, 1214, 1218, 1232, 1262
MD35210, 1203, 1211, 1214, 1215, 1218, 1219, 1232, 1262
MD35212, 1237
MD35220, 276, 279
MD35230, 276, 279
MD35240, 195, 279
MD35242, 274
MD35300, 1189, 1211, 1213, 1214, 1218, 1219, 1232, 1262, 1275
MD35310, 1233, 1258, 1261
MD35350, 1218, 1261
MD35400, 1023, 1252, 1253
MD35410, 1251, 1254
MD35430, 1252
MD35440, 1252
MD35450, 1252
MD35500, 1205, 1209
MD35510, 1209, 1284
MD35550, 1233, 1238
MD35590, 48, 70, 1233, 1247
MD36000, 181, 1216, 1219, 1220, 1291, 1292
MD36010, 83, 84, 181, 1215, 1219, 1220, 1291, 1292
MD36012, 85, 182
MD36020, 84
MD36030, 85
MD36040, 85, 313
MD36042, 313
MD36050, 86, 92
MD36052, 86, 88, 89, 91
MD36060, 48, 49, 1204, 1210, 1284
MD36100, 104
MD36110, 104
MD36120, 104
MD36130, 104
MD36200, 94
MD36210, 93, 346
MD36220, 94
MD36300, 97, 1188, 1189
MD36302, 1213
MD36310, 96, 98
MD36312, 99
MD36400, 81
MD36500, 337
MD36510, 337
MD36600, 103
MD36610, 46, 82, 84, 85, 86, 94, 95, 98, 100, 805
Index

MD36620, 46, 806
MD37002, 301
MD37010, 296
MD37012, 301
MD37020, 298, 308
MD37030, 297
MD37040, 297
MD37060, 296, 299, 300
MD37080, 312
MD51029, 589
MD9004, 321, 595
MD9006, 39, 1556
MD9010, 595
MD9011, 321, 595
MD9424, 595, 702
MD9440, 790
Measuring systems, 337, 1149
Memory requirements
  of basic PLC program, 897
Message signals in DB2, 1066
Mirroring
  Frames, 716
  Retraction direction (lift fast), 567
Modal activation (FOCON/FOCOF), 312
Mode
  AUTOMATIC, 461
  JOG, 462
  JOG in AUTOMATIC, 461
  MDI, 462
Mode change
  from/to the AUTOMATIC, JOG, MDA modes, 467
Mode group, 457, 851
Modes
  change, 466
  -change, 461
  -cross-mode synchronized actions, 462
  of the mode group, 461
  Priorities, 462
Motor/load gear, 343
MTL, 620
Multiinstance DB, 1072
Multitool, 962

NC
  nact, 52
  Languages, 527
  -Failure, 855
  NC Start, 529
  -Read/write variables, 857
  VAR selector, 903
  Input options, 906
  Startup, installation, 914
  NC/PLC interface, 468
  NCK alarm is active, 38
  NCK alarm with processing stop, 38
  NCK battery alarm, 38
  NCK CPU ready, 37
  Negative address extension, 444
  Nonacknowledged gear stage change, 1250
  Non-linear signal distortions, 80
  NORM, 1404

O
  Offset number, 1373
  Operating modes
    Interlocks, 466
    monitoring functions, 466
  Operating states, 463
  Orientation, 132
    tolerance, 232
  Output
    Behavior of an auxiliary function, 401
    -counter, 427
    Sequence, 427
  Overload factor, 187
  Override
    in G331/332, 1331
  OVR, 1340
  OVRA, 1275, 1340

P
  Package
    -counter, 427
  Parameter set
    For axes, 362
    Toolholder with orientation capability, 1446
  Parking, 113, 1153
  Part program
    Channel enable, 530
    Selection, 529
    Skipping of specific part program blocks, 477
    starting, 537
  Password, 62
    resetting, 62
    setting, 62
  Path
    feedrate F, 1308
    velocity, maximum, 1309
Index

Path axes, 671
Path criterion, 190
Path feedrate, 318
PCOF, 1274
Phase filter, 368
Physical quantities, 322
PLC
  - Read/write variable, 57
  - axes, 672
  Basic program functions, 823
  HMI monitor, 893
  Key disable, 884
  Program list, 881
  - Versions, 823
POINTER, 1067
POINTER in FB, 1069
POINTER in FC, 1068
POLF, 1322
POLFMASK, 1322
POLFMLIN, 1322
Polynomials, intersection process, 1437
POS, 671, 672
POSA, 671, 672
Position control loop, 357
Position controller active, 49
Position measuring system, 44
Position of coordinate systems and reference points, 680
Positioning accuracy, 320
Positioning axes, 319, 671
Preprocessing, 467
PRESETON, 686, 1153
PRESETONS, 691
Processing time, 644
PROFIBUS
  - connection, 872
PROFIBUS connection, 873
PROFIBUS diagnostics, 854
PROFINET
  - connection, 875
Program
  - action, 536
  - display status, 533
  Program test, 471
  Runtimes, 641
  - states, 533
Program control, interface signals, 588
Program display modes, 592
Program execution without setpoint outputs, 471
Program operation, 526
  Initial setting, 527
Program section repetition, 540

Programming device
  Hardware requirements, 900
Protection level
  for user ASUB, 572
Protection levels, 60
  programmable, 64
Protection zones, 79, 130
  Data storage, 139
  Definition as per partprogram, 135
  Definition with system variables, 138
  Example, activation, 171
  Example, definition, 162
  Restrictions, 148

R
Radius-related data, 813
Rapid traverse
  - override, 1336
Reaching simulated target point for LEAD with JOG, 520
Recall alarms, 39
Reference axis
  for G96 / G961 / G962, 811
Reference point R, 678
Reference points, 678
Referencing
  in rotary absolute encoders, 1186
  with incremental measurement system, 1157
Referencing methods, 1149
 Relevant standards, 803
Remote diagnostics, 37
Replacement subprogram, 617
replacement zero mark, 1186
REPOS offset, 304
Repositioning neutral axes after SERUPRO, 499
Reset
  behavior, 609
RESET
  Command, 533
Residual time
  For a workpiece, 643
Retraction
  Direction for thread cutting, 1323
Revolutional feedrate (G95), 1311
Rotary axes, 671
Rotary axis parameters, 1446
Rotation component, 771
Rounding, 188
Runtimes
  Program, 641
Run-up completed, 52
Basic Functions
Function Manual, 01/2015, 6FC5397-0BP40-5BA2

Index

S
S functions, 380
S..., 1273
SAVE, 569
SBLOF, 582
SBLOF, 582
SCC, 817
SCPARA, 1357
Screen bright, 38
SD41100, 1312
SD41200, 1268
SD42010, 1319
SD42100, 475, 1341
SD42101, 475, 1342
SD42200, 582
SD42440, 666, 1482
SD42442, 1482
SD42444, 481
SD42665, 193, 232
SD42666, 193, 232
SD42470, 228
SD42475, 228
SD42476, 228
SD42477, 228
SD42480, 1437
SD42496, 1433
SD42500, 251, 252
SD42502, 252
SD42510, 263, 264
SD42512, 264
SD42600, 1282, 1312
SD42676, 232
SD42678, 232
SD42700, 601
SD42750, 592
SD42800, 1269, 1270
SD42900, 1364, 1488, 1489, 1509, 1516, 1533
SD42910, 1486, 1488, 1489, 1509, 1514, 1515, 1516
SD42920, 1488, 1490, 1509, 1514, 1515, 1516
SD42930, 1488, 1490, 1509, 1514, 1516, 1545
SD42935, 1489, 1510, 1515, 1516, 1535, 1540
SD42940, 1478, 1486, 1488, 1491, 1509, 1514, 1517
SD42950, 1469, 1486, 1488, 1509, 1514, 1517
SD42960, 1364, 1489, 1495, 1496
SD42974, 1454
SD42980, 778
SD42984, 1478
SD42990, 590
SD43200, 1279
SD43202, 1279, 1280
SD43206, 1279, 1281
SD43210, 1315, 1317
SD43220, 1315
SD43230, 1315
SD43235, 1289
SD43240, 1207, 1273, 1281
SD43250, 1207, 1282
SD43300, 1282, 1312
SD43400, 109
SD43410, 109
SD43420, 108
SD43430, 108
SD43500, 307
SD43510, 307, 308
SD43520, 308
Secant error, 226
Selecting the cutting edge when changing tool, 1367
Selfacting SERUPRO, 511
SERUPRO
Automatic interrupt pointer, 514
-end ASUB, 431
Programmable interrupt pointer, 512
SERUPRO approach
control from the PLC, 501
SERUPRO ASUB, 304
Special features, 508
seruproMasterChan, 511
Servo gain factor (Kv), 358
SETINT, 559, 568
SETMS, 1272
Setpoint output, 336
Setpoint system, 336
SETTCOR, 1526
Several transverse axes
Axis replacement, 818
Signal distortions, 80
Signal exchange
Cyclic, 821
Event-controlled, 821
Signals
Alarm signals, 37
Axis/spindle-specific (DB31, ...), 36
Channel-specific (DB21, ...), 36
Compile cycles, 838
NCK/PLC, 839
PLC / mode group, 839
PLC/axes, spindles, 841
PLC/NCK, 838
PLC/NCK channels, 840
Ready signals, 37
Simulation, 478
Simulation axes, 338
Single block
  Channel classification, 587
  Program operation mode, 473
SBL1, 581
SBL2, 581
SBL3, 581
Single-axis dynamic response, 1353
Skip levels, 589
Slotting saw, 1392
Smooth approach and retraction
  Significance, 1409
  Sub-movements, 1410
Smoothing
  Path velocity, 210
SOFT, 261
SOFTA, 263
Software limit switch, 104
SPCOF, 1274
SPCON, 1274
Special axes, 670
Special points in the target block
  STOPRRE block, 516
Specify gear stage, 1234
Speed
  feedforward control, 362
  Speed control loop, 357
  Speed controller active, 49
  Speed default, 1279
  Speed setpoint adjustment, 346
  Speed setpoint output, 345
SPI, 1275
spindle
  modes, 1200
Spindle
  -definitions, 1278
  -gear stage 0, 1243
  -interface, 1277
  -job, 1277
  -override, 1336, 1338
  -setpoint speed, 1285
  -speed limits, 1286
  -speed, maximum, 1286
  -speed, minimum, 1285
Spindle disable, 41, 1335
Spindle functions using a PLC, 469
Spindle speed, 319
Spindle speed limitation with G96, G961, 1315
Spline, 179
SPOS, 1205, 1273
SPOSA, 1273
Star/delta switchover with FC17, 1243
Star-delta changeover, 1017
Startup and synchronization of NCK PLC, 850
STOLF, 237
Stop delay area, 556
Stop event, 556
Storing angles in the toolholder data, 1446
Strings, 1074
SVC, 1273
Symbolic programming, 858
Synchronized axes, 673

T
T function, 381, 1366
T function replacement, 620
TCARR, 1458
TCOABS, 1459
TCOFR, 1458, 1459
TCP Tool Center Position, 679
TEACH IN, 462
Thread
  Cutting, 1322
Thread cutting G33, 1317
Tick, 427
TOA
  -Data, 1392
  -Unit, 1368
Tolerance
  -factor for spindle speed, 1285
  with G0, 235
Tool, 1365
  Change, 1366
  Change tool with M06, 1366
  Cutting edge, 1388
  DISC, 1425
  -length, 1395
  management, 457
  -offset data, 1392
  -parameters, 1388
  -retraction, 614
  Select, 1365
  -shape, 1396
  -shape, active, 1399
  -size, active, 1398
  T function, 1366
  Tool base dimension/adapter dimension, 1399
  Tool cutting edge, 1367
  Tool radius compensation 2D (TRC), 1403
  -type, 1389
  -wear, 1398
Tool base dimension, 1399
Index

Tool change
  D function, 1367
  Offset memory, 1368
Tool environments, 1516
Tool length compensation
  calculate tool-specifically, example, 1539
  Geometry, 1395
  Wear, 1398
  Workpiece-specific calculation, 1508
Tool offset
  Offset in the NC, 1370
  Types, 1445
Tool radius compensation
  keep constant, 1434
  Wear, 1398
  Tool radius compensation, 2D, 1402
  Approach and retraction behavior, 1404
  Collision detection, 1430
  Compensation and inner corners, 1428
  Compensation at outside corners, 1424
  Deselection, 1423
  Geometry, 1396
  Modified alarm response, 1437
  Point of intersection G451, 1426
  Selection, 1403
  Smooth approach and retraction, 1409
  Transition circle, 1424
  variable compensation value, 1432
Tool revolver axes, 671
TOOLENV, 1516
Toolholder reference point T, 679
Toolholder selection, 1443
Toolholder with orientation capability, 1364, 1442
  Calculation of active tool length, 1458
  Create new, 1461
  Examples, 1535
  Kinematic chain, 1449
Toolholder, with orientation capability, 1442
  Control system response on Reset, program start, REPOS, 1466
  Create new, 1461
  Inclined machining, 1457
  Programming, 1465
  Supplementary conditions, 1466
  swiveling working table, 1458
Tooth feedrate, 1311
Torsion, 224
TOWBCS, 1513
TOWKCS, 1514
TOWMCS, 1513
TOWSTD, 1512
TOWTCS, 1514
TOWWCS, 1513
TRANS, 664
TRANSMIT, 696
Transverse axis, 811
  Dimensions, 817
Travel to fixed stop, 293
  Block search, 302
  Contour monitoring, 306
  Deselection, 300
  Fixed stop is reached, 297
  Function abort, 306
  Monitoring window, 298
  Positioning axes, 306
  Selection, 296
Traversing movement release, 1186
Traversing ranges, 319

U
  UDT assignments, 859
  Undercut angle, 1400
  User-defined ASUB
    after SERUPRO operation, 495
  UTD-blocks, 858

V
  Velocities, 317
  VELOLIM, 1274
  Vertical axes, 306

W
  WAITENC, 615
  WAITS, 1274
  WALCS0, 112
  WALIMOF, 110
  WALIMON, 110
  WCS, 39
  Work offset
    External work offset, 698
  Working area limitation, 106
    in BCS, 108
    in WCS/SZS, 110
  Working area limitation group, 111
  Workpiece
    -counter, 647
    -simulation, 478
  Workpiece coordinate system (WCS), 39, 663, 703
  Workpiece zero W, 678
Index

X
XE \^ MERGEFORMAT, 378

Z
Zero mark diagnostics, 99
Zero points, 678
Zero vectors, 1449