SIEMENS

# SINUMERIK

# SINUMERIK 828D Commissioning

Commissioning Manual


Preface

Fundamental safety instructions 1

CNC commissioning 2

SINUMERIK Operate 3

Easy XML 4

SINUMERIK Integrate Access MyMachine / OPC UA 5

SINUMERIK Integrate Run MyScreens 6

Appendix A


Valid for:

CNC Software Version 4.7 SP1


01/2015

6FC5397-3DP40-5BA2

# Legal information

## Warning notice system

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

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

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

| ⚠CAUTION |
| --- |
| 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.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

Order number: 6FC5397-3DP40-5BA2
℗ 01/2015 Subject to change

Copyright © Siemens AG 2014 - 2015.
All rights reserved

# 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 under the link (www.siemens.com/motioncontrol/docu):

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

Please send any questions about the technical documentation (e.g. suggestions for improvement, corrections) to the following address: (mailto: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: MDM (www.siemens.com/mdm)

## Training

You can find information about the range of training courses under:

- SITRAIN (www.siemens.com/sitrain) - training courses from Siemens for automation products, systems and solutions
- SinuTrain (www.siemens.com/sinutrain) - training software for SINUMERIK

## FAQs

You can find Frequently Asked Questions in the Service&Support pages under Product Support (www.siemens.com/automation/service&support).

## SINUMERIK

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

## Target group

This documentation is intended for commissioning personnel.

The plant or system is readily assembled and wired. For the following steps, e.g. configuring the individual components, the Commissioning Manual contains all necessary information or at least references.

## Benefits

The intended target group can use the Commissioning Manual to test and commission the system or the plant correctly and safely.

Utilization phase: Setup and commissioning phase

## Standard version

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

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

Country-specific telephone numbers for technical support are provided in the Internet under "Contact" (www.siemens.com/automation/service&support).

## EC Declaration of Conformity

The EC declaration of conformity for the EMC directive can be found in the Internet under: (http://support.automation.siemens.com/WW/view/de/10805517/134200)

# Table of contents


Preface ..... 3
1 Fundamental safety instructions ..... 7
1.1 General safety instructions ..... 7
1.2 Handling electrostatic sensitive devices (ESD) ..... 10
1.3 Industrial security ..... 11
1.4 Residual risks of power drive systems ..... 11
2 CNC commissioning ..... 15
3 SINUMERIK Operate ..... 373
4 Easy XML ..... 737
5 SINUMERIK Integrate Access MyMachine / OPC UA ..... 867
6 SINUMERIK Integrate Run MyScreens ..... 897
A Appendix ..... 1133
A.1 Documentation overview SINUMERIK 828D ..... 1133

# Fundamental safety instructions

# 1

## 1.1 General safety instructions



**⚠DANGER**

**Danger to life due to live parts and other energy sources**

Death or serious injury can result when live parts are touched.

- Only work on electrical devices when you are qualified for this job.
- Always observe the country-specific safety rules.

Generally, six steps apply when establishing safety:

1. Prepare for shutdown and notify all those who will be affected by the procedure.
2. Disconnect the machine from the supply.
   - Switch off the machine.
   - Wait until the discharge time specified on the warning labels has elapsed.
   - Check that it really is in a no-voltage condition, from phase conductor to phase conductor and phase conductor to protective conductor.
   - Check whether the existing auxiliary supply circuits are de-energized.
   - Ensure that the motors cannot move.
3. Identify all other dangerous energy sources, e.g. compressed air, hydraulic systems, or water.
4. Isolate or neutralize all hazardous energy sources by closing switches, grounding or short-circuiting or closing valves, for example.
5. Secure the energy sources against switching on again.
6. Ensure that the correct machine is completely interlocked.

After you have completed the work, restore the operational readiness in the inverse sequence.



**⚠WARNING**

**Danger to life through a hazardous voltage when connecting an unsuitable power supply**

Touching live components can result in death or severe injury.

- Only use power supplies that provide SELV (Safety Extra Low Voltage) or PELV-(Protective Extra Low Voltage) output voltages for all connections and terminals of the electronics modules.



**⚠WARNING**

**Danger to life when live parts are touched on damaged devices**

Improper handling of devices can cause damage.

For damaged devices, hazardous voltages can be present at the enclosure or at exposed components; if touched, this can result in death or severe injury.

- Ensure compliance with the limit values specified in the technical data during transport, storage and operation.
- Do not use any damaged devices.



**⚠WARNING**

**Danger to life through electric shock due to unconnected cable shields**

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the cores of cables that are not used at one end at the grounded housing potential.



**⚠WARNING**

**Danger to life due to electric shock when not grounded**

For missing or incorrectly implemented protective conductor connection for devices with protection class I, high voltages can be present at open, exposed parts, which when touched, can result in death or severe injury.

- Ground the device in compliance with the applicable regulations.

**⚠WARNING**

**Danger to life due to fire spreading if housing is inadequate**

Fire and smoke development can cause severe personal injury or material damage.

- Install devices without a protective housing in a metal control cabinet (or protect the device by another equivalent measure) in such a way that contact with fire is prevented.
- Ensure that smoke can only escape via controlled and monitored paths.

**⚠WARNING**

**Danger to life through unexpected movement of machines when using mobile wireless devices or mobile phones**

Using mobile wireless devices or mobile phones with a transmit power > 1 W closer than approx. 2 m to the components may cause the devices to malfunction, influence the functional safety of machines therefore putting people at risk or causing material damage.

- Switch the wireless devices or mobile phones off in the immediate vicinity of the components.

**⚠WARNING**

**Danger to life due to fire if overheating occurs because of insufficient ventilation clearances**

Inadequate ventilation clearances can cause overheating of components with subsequent fire and smoke. This can cause severe injury or even death. This can also result in increased downtime and reduced service lives for devices/systems.

- Ensure compliance with the specified minimum clearance as ventilation clearance for the respective component.

**⚠WARNING**

**Danger to life when safety functions are inactive**

Safety functions that are inactive or that have not been adjusted accordingly can cause operational faults on machines that could lead to serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

**Note**

**Important safety notices for Safety Integrated functions**

If you want to use Safety Integrated functions, you must observe the safety notices in the Safety Integrated manuals.

**⚠WARNING**

**Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization**

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

## 1.2 Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.



**NOTICE**

**Damage through electric fields or electrostatic discharge**

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

## 1.3 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.

## 1.4 Residual risks of power drive systems

The control and drive components of a drive system are approved for industrial and commercial use in industrial line supplies. Their use in public line supplies requires a different configuration and/or additional measures.

These components may only be operated in closed housings or in higher-level control cabinets with protective covers that are closed, and when all of the protective devices are used.

These components may only be handled by qualified and trained technical personnel who are knowledgeable and observe all of the safety instructions on the components and in the associated technical user documentation.

When assessing the machine's risk in accordance with the respective local regulations (e.g., EC Machinery Directive), the machine manufacturer must take into account the following residual risks emanating from the control and drive components of a drive system:

1. Unintentional movements of driven machine components during commissioning, operation, maintenance, and repairs caused by, for example,
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices/mobile phones in the immediate vicinity of the control system
   - External influences/damage
2. In the event of a fault, exceptionally high temperatures, including an open fire, as well as emissions of light, noise, particles, gases, etc. can occur inside and outside the inverter, e.g.:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage

   Inverters of the Open Type/IP20 degree of protection must be installed in a metal control cabinet (or protected by another equivalent measure) such that contact with fire inside and outside the inverter is not possible.
3. Hazardous shock voltages caused by, for example,
   - Component failure
   - Influence during electrostatic charging
   - Induction of voltages in moving motors
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - External influences/damage

4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

**Note**

The components must be protected against conductive contamination (e.g. by installing them in a control cabinet with degree of protection IP54 according to IEC 60529 or NEMA 12).

Assuming that conductive contamination at the installation site can definitely be excluded, a lower degree of cabinet protection may be permitted.

For more information about residual risks of the components in a drive system, see the relevant sections in the technical user documentation.

SIEMENS

# SINUMERIK

# SINUMERIK 828D
# CNC commissioning

Commissioning Manual


Scope of delivery and requirements 1

Introduction and use of data classes 2

Operating software settings 3

Parameterizing NC machine data 4

Commissioning the drive 5

Optimizing a drive 6

Service Planner 7

Easy Extend 8

Tool management 9

Easy Archive 10

Appendix A


Valid for:

CNC software Version 4.7 SP1


01/2015

6FC5397-3DP40-5BA2

# Legal information

## Warning notice system

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

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

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

| ⚠CAUTION |
| --- |
| 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.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

Order number: 6FC5397-3DP40-5BA2
℗ 03/2015 Subject to change

Copyright © Siemens AG 2009 - 2015.
All rights reserved

# Table of contents


**1 Scope of delivery and requirements ..... 9**

1.1 System overview ..... 9

1.2 Commissioning and service tools ..... 11

1.3 Commissioning process ..... 13
1.3.1 Guide to the documentation ..... 13

1.4 Starting up the control ..... 16

1.5 Communication with the control ..... 19
1.5.1 Communicating with the control via X130 ..... 19
1.5.2 How to communicate with the control using the Programming Tool ..... 21
1.5.3 How to communicate with the control via Access MyMachine ..... 22

**2 Introduction and use of data classes ..... 25**

2.1 Data classes in the NCK ..... 26

2.2 Data classes in the PLC ..... 29

2.3 Data classes in the operating software ..... 30

**3 Operating software settings ..... 33**

3.1 Access levels ..... 33

3.2 How to set and change the password ..... 35

3.3 How to set the date and time ..... 36

3.4 Setting the operating software languages ..... 37
3.4.1 Available system languages and language extensions ..... 37
3.4.2 This is how you delete a language extension ..... 38
3.4.3 Entering Asian characters using the Input Method Editor ..... 38
3.4.4 This is how you enter the Chinese character ..... 40
3.4.5 This is how you enter Korean characters ..... 41

3.5 Checking and entering licenses ..... 43
3.5.1 How to enter a license key ..... 43
3.5.2 This is how you determine missing licenses/options ..... 45
3.5.3 Definitions for license management ..... 46

3.6 Configuring PLC alarms ..... 48
3.6.1 Structure of the PLC alarms ..... 48
3.6.2 How to create PLC alarms ..... 50
3.6.3 Configuring the alarm log ..... 51
3.6.4 How to configure the log ..... 52

3.7 Creating OEM-specific online help ..... 55
3.7.1 Structure and syntax of the configuration file ..... 56
3.7.2 Structure and syntax of the help book ..... 57
3.7.3 Description of the syntax for the online help ..... 59
3.7.4 Example: This is how you generate an OEM-specific online help book ..... 62

3.7.5 Example: How to create an online help for user PLC alarms ... 65
3.7.6 Example: This is how you generate an online help for NC/PLC variables ... 68
3.7.7 Example: This is how you generate a programming online help ... 70

**4 Parameterizing NC machine data ... 73**

4.1 Classification of machine data ... 73
4.2 Processing part programs from external CNC systems ... 76
4.3 Machining free form surfaces using Advanced Surface ... 77
4.4 Machine data for Advanced Surface ... 78
4.5 Machine data for an analog spindle ... 81
4.6 Locking the control (CNC lock function) ... 84
4.7 Managing data ... 86
4.7.1 How to transfer data within the controller ... 87
4.7.2 To save and load data ... 88
4.7.3 How to compare data ... 89
4.8 Friction compensation with adaptive characteristics ... 90
4.8.1 Friction compensation with velocity injection pulse ... 91
4.8.2 Friction compensation with torque injection pulse ... 93
4.8.3 Data list ... 94
4.8.4 Example: Friction compensation for a linear axis ... 94

**5 Commissioning the drive ... 97**

5.1 Commissioning the drive ... 97
5.1.1 Sample configuration – overview ... 97
5.1.2 Automatic first commissioning ... 99
5.1.3 Example: This is how you check the infeed configuration (S120 Combi) ... 103
5.1.4 Example: This is how you configure the spindle (DO3) ... 104
5.1.5 Example: This is how you configure an encoder for an axis (DO4) ... 114
5.1.6 Example: This is how you check the configuration of an axis (DO6) ... 119
5.1.7 Example: This is how you configure the infeed (S120 Booksize) ... 123
5.1.8 Replacing a SINAMICS S120 component ... 130
5.1.9 Example: Parallel connection with TM120 ... 131
5.1.10 Exchanging process data ... 133
5.2 Assigning axes ... 134
5.2.1 Example: This is how you assign the axes ... 134
5.2.2 Example: Setting machine data for an axis/spindle ... 136
5.3 Commissioning auxiliary axes ... 139
5.3.1 Sample configuration – overview ... 139
5.3.2 Example: How to configure an auxiliary axis ... 141
5.4 Topology rules for DRIVE-CLiQ ... 144
5.4.1 Topology rules for S120 Combi ... 144
5.4.2 Topology rules for S120 Booksize ... 146
5.4.3 Topology rules for SMC40 ... 150
5.5 Terminal assignments ... 151
5.5.1 Terminal assignment at X122 and X132 ... 151
5.5.2 Terminal assignment at X242 and X252 ... 152
5.5.3 Example: Circuitry for a CU with line contactor ... 154

5.5.4 Inputs/outputs (dialog) ....156
5.5.5 Interconnections (dialog) ....158
5.5.6 Connecting the probes ....159
5.6 Configuring data sets ....161
5.6.1 Data sets - overview ....161
5.6.2 Adding a data set ....162
5.6.3 Remove data set ....168
5.6.4 Modify data set ....171
**6 Optimizing a drive ....173**
6.1 Control response ....173
6.2 Automatic servo optimization ....176
6.2.1 Options before the start ....176
6.2.2 Selecting the tuning strategy ....178
6.2.3 How to start the automatic servo optimization ....183
6.2.4 This is how you start path interpolation ....186
6.3 Optimizing the axis jerk ....189
6.3.1 Checking the axis jerk ....189
6.3.2 Parts program for axis jerk ....190
6.3.3 Selecting trace signals ....191
6.3.4 Optimizing the jerk setting ....194
6.4 Torque load ....197
6.5 Optimize the spindle ....201
6.5.1 Set the machine data for the spindle ....201
6.5.2 Spindle: Checking the speed controller ....202
6.5.3 Spindle: Checking the position controller ....208
6.6 Circularity test ....211
6.6.1 Circularity test: Function ....211
6.6.2 Circularity test: Performing the measurement ....212
6.6.3 Circularity test: Examples ....214
6.6.4 Circularity test: Saving data ....217
**7 Service Planner ....221**
7.1 Interfaces in the PLC user program ....223
7.2 Interfaces to the operating software ....228
7.3 This is how you import and export maintenance tasks ....231
7.4 Acknowledging maintenance tasks ....234
**8 Easy Extend ....235**
8.1 Function of Easy Extend ....235
8.2 Configuration in the PLC user program ....237
8.3 Option bits for machine manufacturers and dealers ....239
8.4 Display on the user interface ....241
8.5 Generating language-dependent text ....242
8.6 Examples ....244

8.6.1 Example with parameters to support the commissioning ... 244
8.6.2 Example with control elements ... 246
8.6.3 Example of a power unit ... 247

**9 Tool management ... 249**

9.1 Fundamentals ... 249
9.1.1 Tool Management - Overview ... 249
9.1.2 Structure of the tool management ... 250
9.1.3 Components of the tool management ... 251
9.1.4 Multitool ... 254
9.1.5 Manual tools ... 255

9.2 Machine data for the tool management ... 257

9.3 NC ↔ PLC user interface ... 262
9.3.1 Relocating, unloading, loading tool, positioning magazine ... 264
9.3.2 Tool change ... 272
9.3.3 Transfer-step and acknowledgment-step tables ... 281

9.4 PLC Program Blocks ... 284
9.4.1 Acknowledgment process ... 284
9.4.2 Types of acknowledgment ... 285
9.4.3 Acknowledgment states ... 286
9.4.4 Configuring step tables ... 291
9.4.5 Configuring acknowledgment steps ... 293
9.4.6 Adjust the PLC user program ... 295
9.4.7 Information on magazine location ... 296
9.4.8 PI service: TMMVTL ... 298

9.5 Example: Loading/unloading ... 300

9.6 Example: Change manual tools ... 302

9.7 Application example for turning machine ... 306
9.7.1 Example: Turning machine with revolver magazine (MAGKONF_MPF) ... 306
9.7.2 Example: Acknowledgment steps (turning machine) ... 311
9.7.3 Example: Tool change cycle for turning machine ... 312
9.7.4 Example: Block search and MCALL ... 315
9.7.5 Example: Turning machine with counterspindle ... 315
9.7.6 Example: Test for empty buffer ... 316
9.7.7 Example: Transporting a tool from a buffer into the magazine ... 316
9.7.8 Example: Repeat "Prepare tool change" order ... 317

9.8 Application example for milling machine ... 319
9.8.1 Example: Milling machine with chain magazine and dual gripper (MAGKONF_EXAMPLE_MPF) ... 319
9.8.2 Flow chart: Tool change ... 324
9.8.3 Example: Acknowledgment steps (milling machine) ... 333
9.8.4 Example: Tool change cycle for milling machine ... 335

**10 Easy Archive ... 337**

10.1 Backing up and archiving data ... 337

10.2 Which data is backed up in the archive? ... 338

10.3 This is how you back up just the machine data that have changed ... 341

10.4 This is how you create a commissioning archive ....342
10.5 This is how you import a commissioning archive ....344
10.6 Example: Data archiving "Easy Archive" (use case) ....345
10.7 Parameterizing the serial interface ....347
**A Appendix ....349**
A.1 Table: Language code in file names ....349
A.2 List of abbreviations ....350
A.3 Information about third-party software used ....353
**Index ....355**

# 1 Scope of delivery and requirements

## 1.1 System overview

### Configuration with SINAMICS S120 Booksize

The following configuration shows a typical example with SINAMICS S120 booksize:

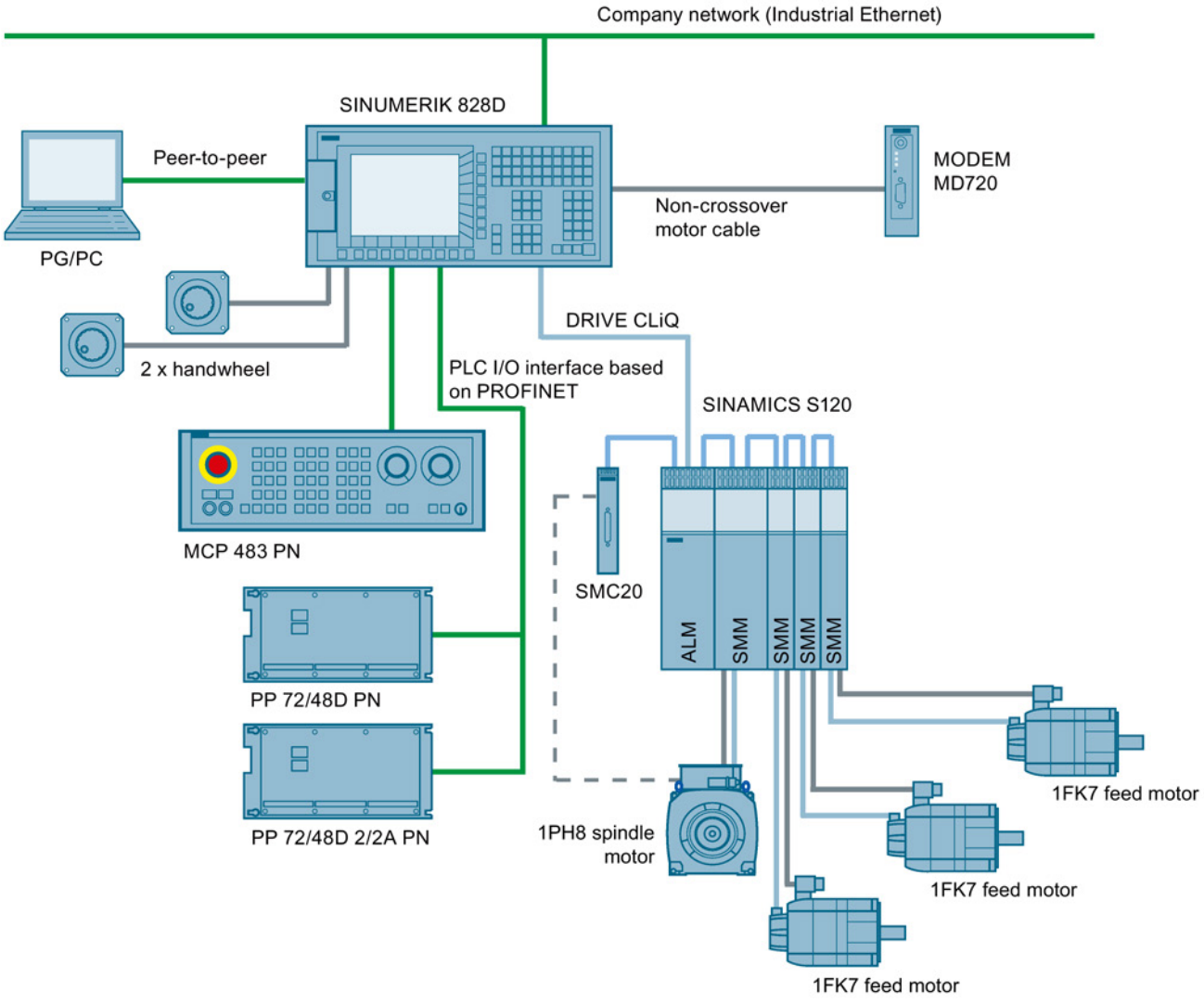



Figure 1-1 Configuration example 1: Basic configuration with 4 axes

## Configuration with SINAMICS S120 Combi

The following configuration shows an example with SINAMICS S120 Combi:

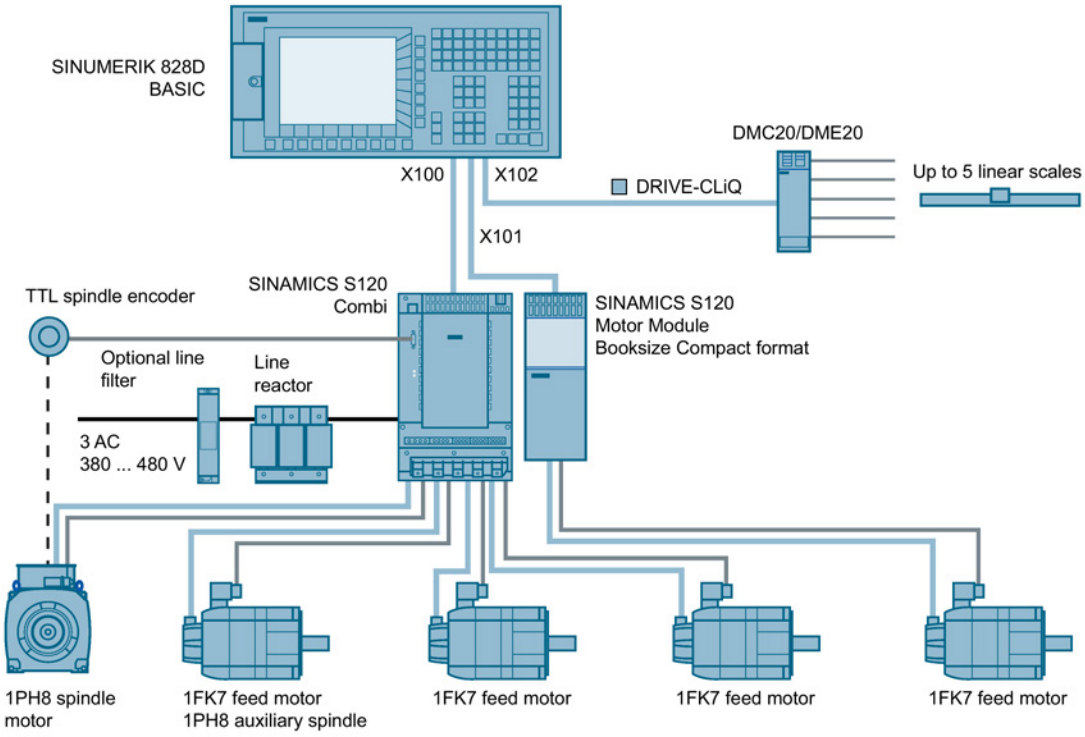



Figure 1-2 Configuration example 2: SINUMERIK 828D BASIC with S120 Combi

# 1.2 Commissioning and service tools

## Toolbox CD

The Toolbox CD for SINUMERIK 828D has the following content:

- Configuration data
- PLC Programming Tool
- Access MyMachine /P2P
- Program for cycle protection
- Examples
- Licenses

## PLC Programming Tool

The following tool is available for programming the PLC: PLC Programming Tool for Integrated PLC. The designation "Programming Tool" is used in the rest of this manual.

## SINUMERIK Integrate Access MyMachine /P2P

Access MyMachine / P2P is a tool that allows the commissioning engineer to exchange files between the PC and the control very easily.

For data transmission, the PC is connected directly to the Ethernet interface on the front of the control. For a point-to-point connection, time-consuming parameterization of the Ethernet interface is not necessary. All settings are made automatically by Access MyMachine /P2P. Several NCUs can be accessed in sequence via a company network with Access MyMachine /P2P.

The designation "Access MyMachine" is used in the rest of this manual.

## SINUMERIK Integrate Create MyConfig

Create MyConfig helps the machinery manufacturer to create and execute a project to automate machine commissioning. Even upgrading CNC systems at end users can be configured and executed automatically. This has the following advantages, for example:

- The time required for commissioning or upgrade is significantly reduced
- Errors are avoided when commissioning or upgrading through structured preparation and automation of process steps
- Reproducibility of the automatic commissioning or upgrading

The description of this tool is supplied together with the software on CD.

## STARTER drive/commissioning software

Drive commissioning for the SINUMERIK 828D can be performed using the STARTER drive/commissioning software.

- Simple commissioning procedures which are usually performed by field service staff (such as activating direct measuring systems) are executed directly via the SINUMERIK 828D user interface.
- Advanced commissioning procedures which are usually performed when the machine is being manufactured can also be executed offline via the commissioning software used for SINAMICS S120.

# 1.3 Commissioning process

## 1.3.1 Guide to the documentation

### Sequence

The commissioning process described in the following table represents an efficient suggestion that is also discussed in the SINUMERIK training. The tasks shown in square brackets apply to the numbers of the steps described in the training documents. The sections contained in the manual that provide supplementary information to the training documents are specified as references in the right-hand column.

### Overview

| | | |
|---|---|---|
| **<B060>** | System overview | "Commissioning CNC" (Page 9)<br>Commissioning Manual |
| **<B002>** | Commissioning and service tools | "Commissioning CNC" (Page 11)<br>Commissioning Manual |

### Basic information

| | | |
|---|---|---|
| <B061> | Menu structure | Measuring Cycles Programming Manual |
| **<B021>** | Access levels | "Commissioning CNC" (Page 33)<br>Commissioning Manual |
| **<B021>** | Date and time | "Commissioning CNC" (Page 36)<br>Commissioning Manual |
| **<B041>** | Parameterizing machine data | "Commissioning CNC" (Page 73)<br>Commissioning Manual |
| <B032> | SINUMERIK Integrate Access MyMachine | Online Help |
| **<B044>** | Checking and entering licenses | "Commissioning CNC" (Page 43)<br>Commissioning Manual |
| <B045> | Network configuration → define drives | Turning/Milling/Grinding/Universal Operating Manual |
| <B007> | Electronic logbook | "SINUMERIK Operate"<br>Commissioning Manual |
| **<B017>** | Data management | "Commissioning CNC"<br>Commissioning Manual<br>• Introduction and use of data classes (Page 25)<br>• Easy Archive (Page 337) |
| <B006> | Electrocabinet structure | "EMC Guidelines"<br>Configuration Manual |

## Configuration and diagnostics

| | |
|---|---|
| <B010> PPU connections diagnosis<br>• System overview<br>• Digital inputs/outputs | "PPU and Components" Manual |
| **<B030>** Digital inputs/outputs → terminal assignment<br>Handwheels and measuring inputs | "Commissioning CNC" (Page 151)<br>Commissioning Manual |
| <B051> Line Modules, Motor Modules | SINAMICS S120 Device Manual |
| <B053> Diagnosis<br>• Drive system diagnostics<br>• Diagnostics and service → axes | "SINUMERIK Operate"<br>Commissioning Manual |
| <B008> PROFINET connections diagnosis:<br>• MCP 310C PN / MCP 483C PN<br>• MCP 310 USB / MCP 483 USB<br>• PP 72/48D PN and<br>PP 72 / 48D 2/2A PN | "PPU and Components" Manual |

## PLC

| | |
|---|---|
| <B026> Creating a PLC program | "PLC Programming Tool"<br>Online Help |
| <B019> S7-200 PLC instructions | "PLC Programming Tool"<br>Online Help |
| <B033> PLC functions | "Fundamentals" Function Manual (P4) |
| <B031> PLC alarms and messages → function interface: PLC user alarms | "Fundamentals" Function Manual (P4) |
| <B058> PLC onboard diagnosis | Turning/Milling/Grinding/Universal Operating Manual<br>"Fundamentals" Function Manual (P4) |
| <B000> PLC interface structure | "NC Variables and Interface Signals"<br>List Manual, Section: "Interface signals – overview" |

## Commissioning process

| | |
|---|---|
| **<B068>** Commissioning process → placing a drive into operation | "Commissioning CNC" (Page 97)<br>Commissioning Manual |
| <B084> Homing encoder adjustment | "Fundamentals" Function Manual (R1) |
| **<B101>** Drive tuning | "Commissioning CNC" (Page 173)<br>Commissioning Manual |
| **<B102>** Tool management | "Commissioning CNC" (Page 249)<br>Commissioning Manual |
| **<B009>** Easy Extend | "Commissioning CNC" (Page 235)<br>Commissioning Manual |
| <B0 ... > Easy XML | "Easy XML"<br>Programming Manual |
| <B011> SINUMERIK Integrate Run MyScreens | "SINUMERIK Integrate Run MyScreens"<br>Programming Manual |
| **<B020>** Service Planner | "Commissioning CNC" (Page 221)<br>Commissioning Manual |
| <B018> Easy Message | Turning/Milling/Grinding/Universal Operating Manual |
| <B016> Installation of an additional axis | --- |
| **<B029>** Analog spindle | "Commissioning CNC" (Page 81)<br>Commissioning Manual |

## General conditions

This manual assumes the following general conditions:

- The mechanical and electrical installation of the system must have been completed.
- Check the system visually for:
  - Correct mechanical installation with secure electrical connections
  - Connection of the power supply
  - Connection of shielding and grounding
- Switching on the control and startup in "Normal startup":

  The control has finished booting when the main screen of the operating software is displayed.

## References

Detailed information on cycle startup under SINUMERIK Operate can be found in the SINUMERIK Operate 828D Commissioning Manual, Section "Configuring cycles".

# 1.4 Starting up the control

## Control startup

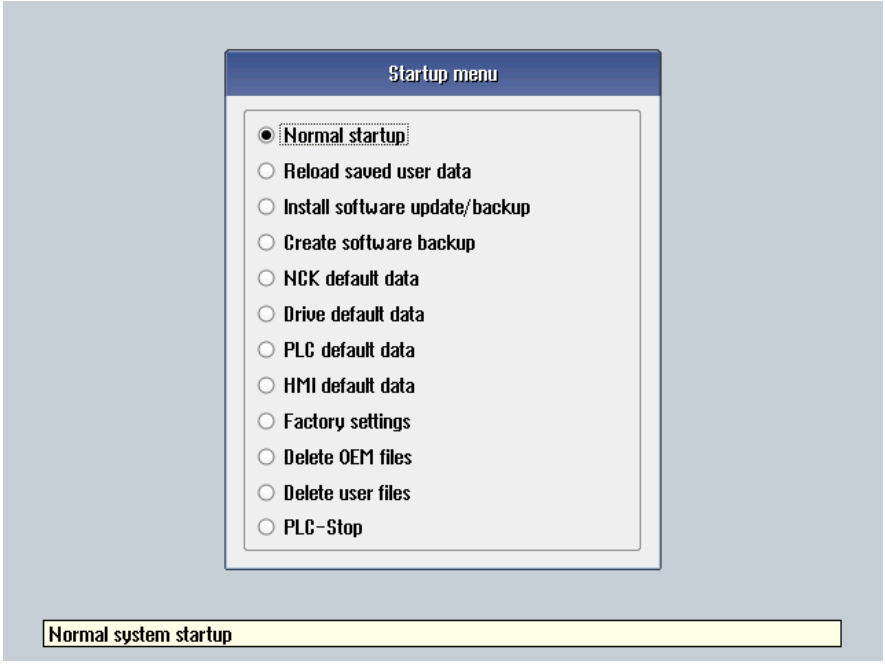
Procedure:

1. Switch the control on. The following display then appears during startup:




2. Press the <SELECT> key within three seconds.
3. Press the following keys in succession:

   Menu reset key, **HSK2** (horizontal SK2), **VSK2** (vertical SK2)
4. The "Startup menu" is displayed, "Normal startup" is the default setting.




## Operating modes for startup

| Selection | Function |
|---|---|
| Normal startup | The system carries out a normal startup. |
| Reload saved user data | The system loads the stored user data ("Save data" softkey) from the system CompactFlash card. |
| Install software update/backup | An update is installed on the system CompactFlash card from the user CompactFlash card or USB FlashDrive. |
| Create software backup | A backup of the system CompactFlash card is saved to the user CompactFlash card or USB FlashDrive. |
| NCK default data | The system loads the Siemens NCK data default settings and deletes the retentive data on the PLC. |

| Selection | Function |
|---|---|
| Drive default data | The SINAMICS user data is deleted. |
| PLC default data | PLC general reset and load default NOP PLC program. |
| HMI default data | Choice between two cases: **No** [case 1] **/ Yes** [case 2]<br>• **Case 1:**<br>User data in the /user directory is deleted.<br>• **Case 2:**<br>As case 1 and additionally:<br>Deletion of the data in the /oem and /addon directories - except the logbook data. |
| Factory settings | Choice between two cases: **No** [case 1]**/ Yes** [case 2]<br>• **Case 1:**<br>The SINAMICS user data is deleted.<br>Siemens standard NCK data is loaded.<br>PLC general reset and load default NOP PLC program.<br>User data in the /user directory is saved.<br>• **Case 2:**<br>As case 1 and additionally:<br>Deletion of the data in the /oem and /addon directories - except the logbook data. |
| Delete OEM files | All additional machine manufacturer data is deleted. |
| Delete user files | |
| PLC stop | The PLC is stopped. |

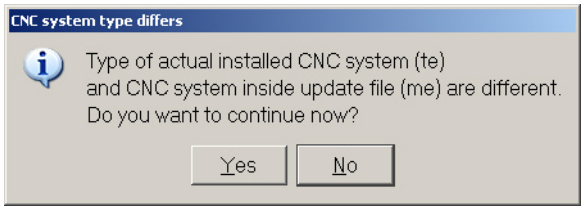
## Selecting the grinding technology

During the control startup, the startup menu provides the option to load the cylindrical grinding or the surface grinding configuration.

To switch the grinding technology, the grinding technology is selected again in the startup menu. The default data is then loaded and the desired technology scaled. The licenses and options are retained.

## Checking the hardware

The technology is determined during startup for the PPU 2xy.3 hardware. This distinction is made using the article number found in the hardware information block, which is then compared with the existing NCK scaling. The following message is output if software that is not suitable for this hardware is found:




**Note**

**Replacement of the system CompactFlash card between different PPUs**

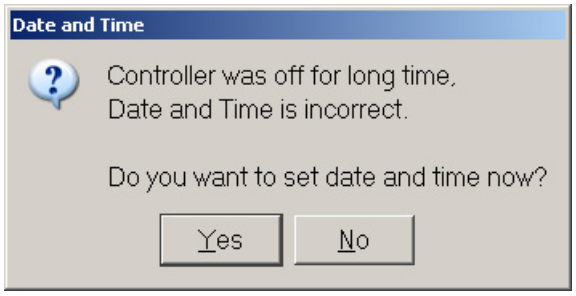
Because of the system-related dependency between the CompactFlash card and SRAM for the data storage in the SINUMERIK 828D, the system CompactFlash card should be considered as a permanently installed EEPROM and should not be replaced!

If this has to be performed for imperative reasons, the replacement of the system CompactFlash card is detected during startup because of the stored serial number.

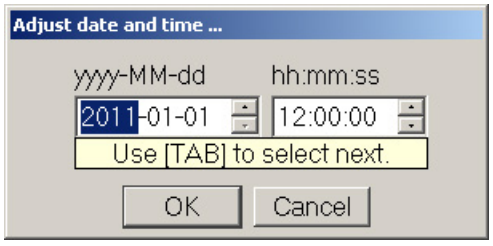
The reaction of the control is the loading of saved data during startup (backup was performed previously with "Save data" softkey). If no stored data is found, a startup is performed automatically with the "NCK default data".

## Empty RTC capacitor

If the RTC capacitor is discharged, the following message is issued during startup:




You can then reset the date and time:




The capacitor is then charged again when the control is switched on during startup.

# 1.5 Communication with the control

### Creating the connection

An Ethernet cable is needed to connect the control and PG/PC. The following Ethernet interfaces are available on the control:

- **Connection via X127** (behind the flap on the front):

  Cable type: Crossed Ethernet cable

  At interface X127, the control is preset as a DHCP server, delivering the IP address192.168.215.1 for a direct connection (peer-to-peer connection).

- **Connection via X130** (at the back):

  Cable type: Uncrossed Ethernet cable

  The interface X130 is the connection to the company network. The IP address that the PG/PC receives here as a DHCP client is determined by the DHCP server from the company network or fixed IP address is entered manually.

## 1.5.1 Communicating with the control via X130

### Connection to the company network

The NCU is connected to the company network via the Ethernet interface X130. The company network is used, for example, to access the network drives.

In the "Diagnostics" operating area select the "Bus TCP/IP " → "TCP/IP Diagnostics" → "Details" softkey with the menu forward key in order to set the parameters for the communication via X130.

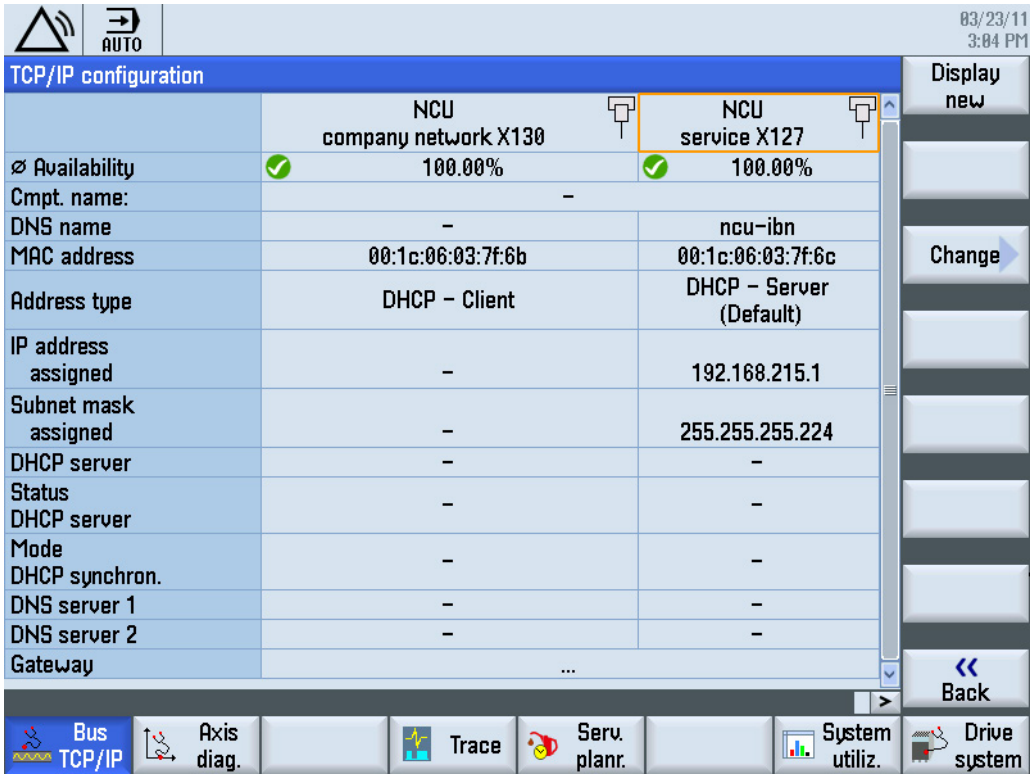



Figure 1-3 Network settings

### Company network X130

| | | |
|---|---|---|
| | white | Network cable inserted |
| | red | Network cable not inserted |

### Availability

The availability describes the percentage of faulty data compared to the entire data volume. Problems in the company network (e.g. logical drives that cannot be reached, double IP address, etc.) as well as settling time during power up can result in fluctuations in availability:

| | | |
|---|---|---|
| | green | Greater than 95% |
| | yellow | 50 - 95 % |
| | red | Less than 50% |

### Note

All information that is not available is marked in the relevant table line with a hyphen "-".

## 1.5.2 How to communicate with the control using the Programming Tool

### Setting up the communications interface in the PLC Programming Tool

Procedure:

1. Start the PLC Programming Tool via the following link or via the "Start" menu:



2. In the navigation bar, click the "Communication" icon or select "View" → "Communication" from the menu.
3. In the left column, under "Communications parameters" enter 192.168.215.1as the IP address for X127.
4. Double-click the "TCP/IP" icon at the top right.
5. In the dialog "PG/PC interface" select the TCP/IP protocol of the PG/PC. Normally this is the network card of the PC.

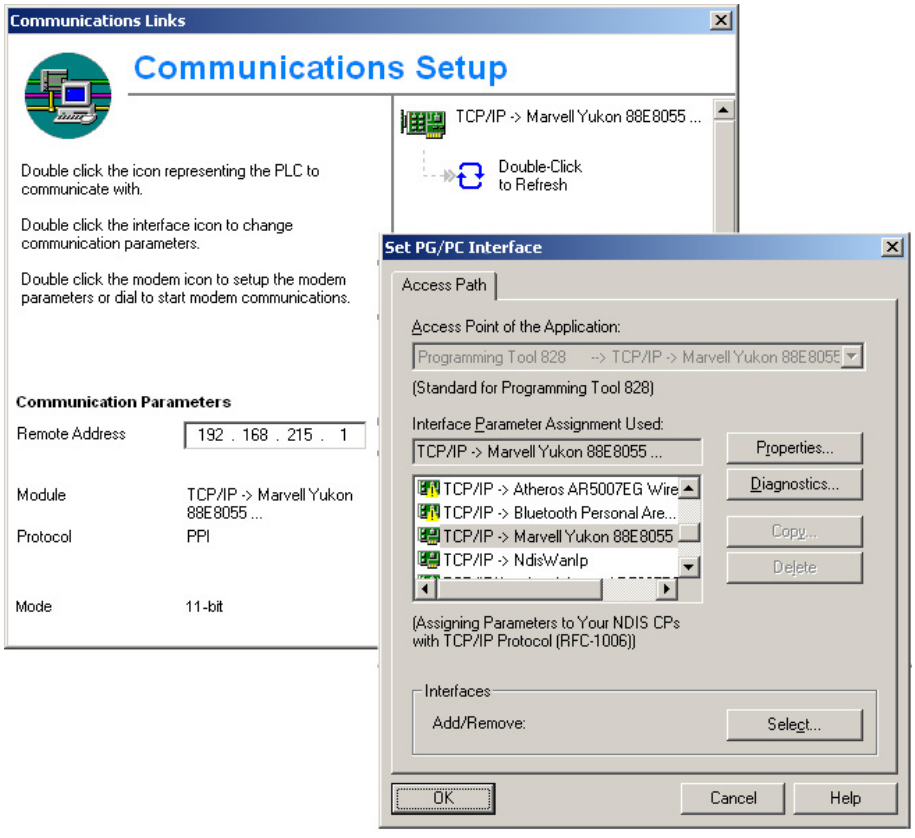



Figure 1-4 TCP/IP communications settings

6. Confirm with "OK".

7. Connect by double-clicking the "Double-click to update" icon. If the connection is made successfully, the icon will be displayed with a green border:

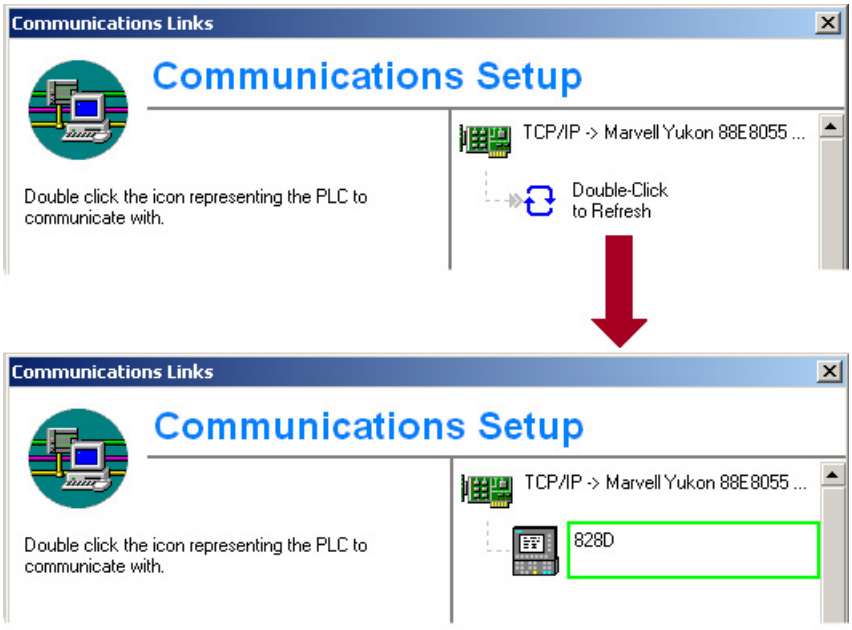



Figure 1-5 Online connection

If a connection is not established, check the settings under "Network connections" → "Local Area Connection".

8. Confirm with "OK" and repeat Step 7.

## 1.5.3 How to communicate with the control via Access MyMachine

### Connection options

The following options are available for "Access MyMachine" to establish a connection to the control.

- Direct connection (peer-to-peer)
- Network connection

The current status of the connection is displayed at the bottom in the status bar of Access MyMachine.

Meaning of the buttons:

 Connect

 Disconnect

 Remote control

**Note**

Generally only **one connection** is permitted, i.e. several simultaneous connections to different controls are not supported: Data exchange between two controls by means of "Access MyMachine" is therefore not possible.

## Direct connection

To create a direct connection:

1. The login data is entered in the dialog "Settings" → "Connection" → "Direct connection":

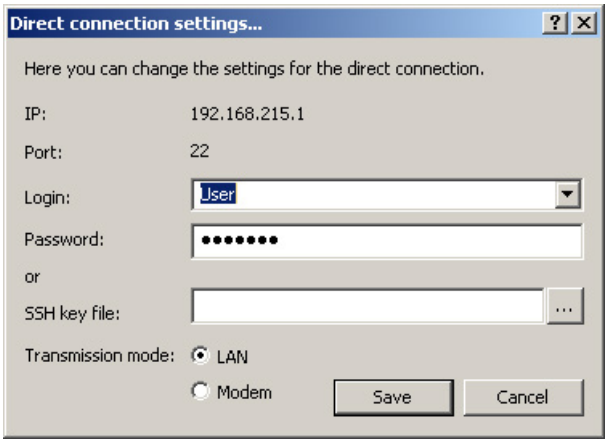



Figure 1-6 Dialog: Login data for direct connection

2. In the menu, select "Connection" → "Connect" → "Direct connection" or click the "Connect" button.

   The following dialog box is displayed:

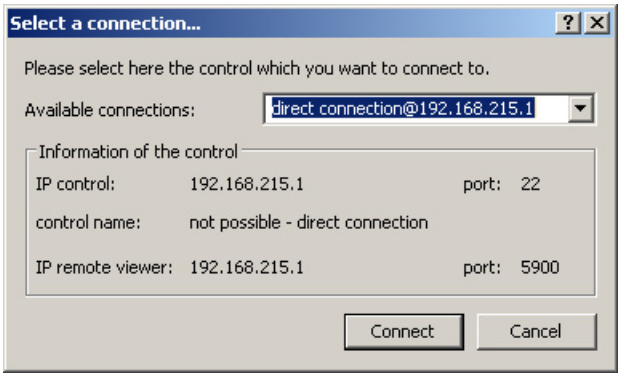



Figure 1-7 Dialog: Direct connection

3. The last selected direct connection is highlighted. Using the "Connect" button, a connection to the IP address196.168.215.1 is created.

   This dialog does not appear when the direct connection is selected using the menu.

## Network connection

To create a network connection:

1. In the menu, select "Settings" → "Connection" → "Direct connection" or click the "Connect" button.

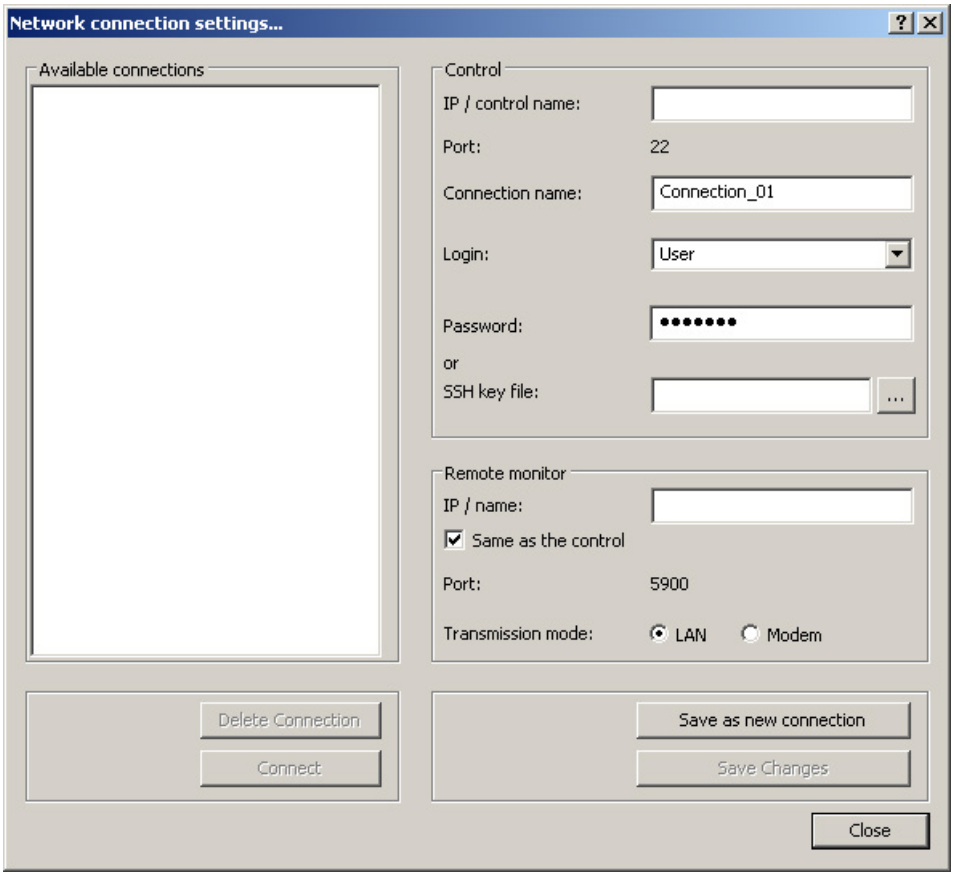



Figure 1-8 Dialog: Network connection

2. In the menu, select "Connection" → "Connect" → "Network connection" or select – if available – one of the previously selected connections.
3. Connection is made to the parameterized control.

**Note**

**SSh key file**

As an alternative to entering a password, the user may also use an SSh key for authentication. Please refer to the Online Help for more information on this topic.

# Introduction and use of data classes

# 2

## Objective

The principle of data classes creates a clear demarcation between system data, (data that is created when installing the software), OEM data (data that is created when commissioning a prototype machine) and user data (data that is created by the user). OEM data is further classified into data that is valid for a series of machines (standard data for all machines of a series) and individual data, which only applies to a specific machine.

This data classification is realized when creating data class archives, i.e. archives that only include a subset of data (manufacturer data, individual machine data or user data). This simplifies commissioning and upgrading.

# 2.1 Data classes in the NCK

## Classification

All control-relevant data is classified into data classes regarding its later use in commissioning, software upgrade, software update and component replacement. Data is classified according to four data classes: M (Manufacturer), I (Individual), U (User) and S (System)

## System (S) data class

This data class covers data in the Siemens and System directories on the CompactFlash Card. For the rest of this document it will be referred to as "S".

## Characteristics / properties of data class S

The data is automatically loaded the first time the system is switched on or during initialization and activated as far as is necessary.

"S" class data cannot be backed up and are designated as "read only".

| Data type | Example |
|---|---|
| HMI: | Siemens standard user interface and additional languages available |
| NCK: | Standard cycles, measurement cycles and JobShop cycles, definitions such as SGUD and SMAC and all NC active data with data class S |
| PLC: | Hardware configurations |
| Drives: | Code and data (data descriptions, alarm descriptions, macros) |

## Manufacturer (M) data class

This data class covers all data defined by the machine manufacturer (OEM) when a machine in a model series is commissioned for the first time. For the rest of this document it will be referred to as "M".

## Characteristics / properties of data class M

Class M data is stored in a separate "Manufacturer" archive.

| Data type | Example |
|---|---|
| HMI: | OEM dialogs (Easy Screen) and cycle support, alarm and message texts. |
| NCK: | Manufacturer cycles, MGUD and MMAC definitions and all NC active data with data class M (no memory normalizing machine data) |
| PLC: | PLC program, contents of the data blocks in this data class |
| Drives: | all |

### Individual (I) data class

This data class covers data concerning a particular machine and is also created during commissioning by the OEM, or later by the dealer. This data class will be referred to as "I" in the rest of this document.

### Characteristics / properties of data class I

Class I data is stored in a separate INDIVIDUAL archive.

| Data type | Example |
|---|---|
| HMI: | Easy Extend and Service Planner |
| NCK: | Reference point dimensions, backlash compensation, compensation data, tool carrier data, all NC active data with data class I |
| PLC: | Contents of the data blocks in this data class |
| Drives: | No data available at this time. |

### User (U) data class

This data class covers all user data as well as data that is created during the machine runtime, for example the maintenance interval timer. This data class will be referred to as "U" in the rest of this document.

### Characteristics / properties of data class U

Class U data is stored in a separate "User" archive.

| Data type | Example |
|---|---|
| HMI: | --- |
| NCK: | Tool data, setting data, part programs, subprograms and user cycles, UGUD and UMAC definitions, but no program code (e.g. PLC) |
| PLC: | Contents of the data blocks in this data class |
| Drives: | No data available at this time. |

### GUD and macros

The following table shows how the definition files (GUD, macros) are assigned to a data class:

| Definition file | Data class |
|---|---|
| GUD4 | Individual (I) |
| MGUD | Manufacturer (M) |
| MMAC | Manufacturer (M) |
| UGUD | User (U) |
| UMAC | User (U) |

## Data class attribute

The data class attribute is set implicitly. The default setting of the data class attribute can be explicitly changed.

- In general, data from the active file system is assigned to a **data class** as it belongs to a certain file.

  Example: Compensation data (data from CEC, EEC or QEC) is assigned to the "Individual" data class. If data are generated as a result of a definition (GUDs and macros) then this data also inherits the data class in which the definition file is located. GUD from UGUD.DEF are assigned the "User" data class.

- However, if the data definition and data content are in **different data classes** , then this must be specified in the data definition.

  Example: The definition of the GUD, which defines a probe, should be located in the "Manufacturer" data class, as it is required to run the manufacturer cycles.

  However, the value of the data should belong to the "Individual" data class as the probe type can differ from machine to machine. This is the reason that the additional variable attribute "DC" (DataClass) has been introduced:

```
MGUD.DEF ("Manufacturer" data class)

DEF CHAN DCI INT CALIPER
```

The definition of the data class of the data content can only be "less than or equal to" the data class of the definition. Any other configurations are rejected with an alarm.

For this purpose, the following weighting of the data classes is assumed: **M > I > U**

# 2.2 Data classes in the PLC

## Data classes in the PLC

Assigning program blocks in data classes is used to efficiently upgrade or commission a SINUMERIK control. Here, a distinction is made between three data classes:

- **Manufacturer (M)** is allocated by the machine manufacturer during construction.
- **Individual (I)** is allocated by the machine manufacturer during the first commissioning (values for this specific machine).
- **User (U)** is set by the end customer (values associated with the particular process). You cannot assign a different data class to the main program or to subroutines.

| Block | Default setting | Can be changed |
|---|---|---|
| Main program (OB1) | MANUFACTURER | No |
| Subroutine (SBR, INT, ...) | MANUFACTURER | No |
| Data block | INDIVIDUAL<br>MANUFACTURER<br>USER | Yes |

## Example

The data classes are assigned to the block in the dialog "Properties":

1. In the menu,"View" > "Data block", select a data block.
2. In the menu "View" → select "Properties" to change the data class.

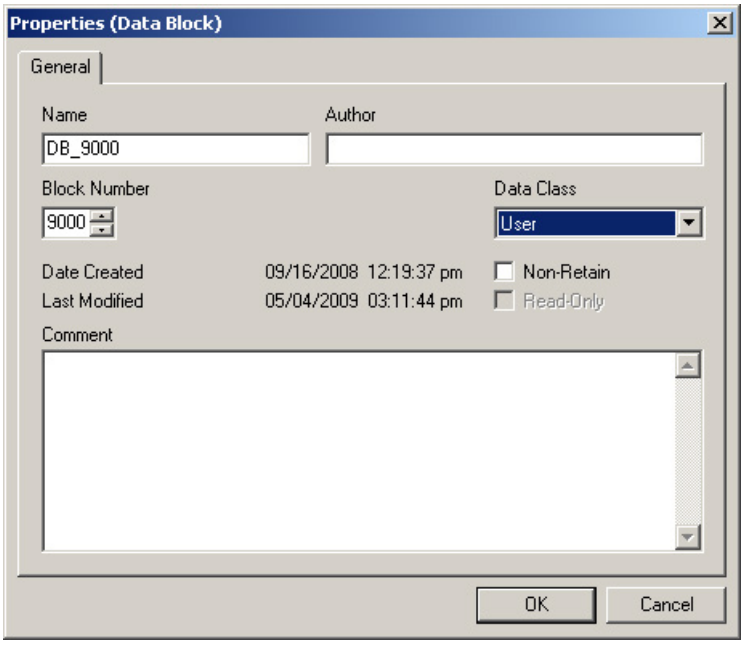



Figure 2-1 Selecting the data class

# 2.3 Data classes in the operating software

## Representing the data classes in the operating software

The arrangement in data classes is essentially already given by the directory structure of the CompactFlash card:

| Directory | Data class |
|---|---|
| system<br>siemens | SYSTEM |
| addon | MANUFACTURER |
| oem | MANUFACTURER, INDIVIDUAL |
| individual | MANUFACTURER, INDIVIDUAL |
| user | USER |

The directories "System" and "Siemens" have no significance for archiving, since they are set up with the installation of the SINUMERIK software and are not changed by installation, configuration or in later use. An update or upgrade of the system will typically be carried out in these directories. The archiving of these directories is for this reason only necessary as a rollback backup in the background.

Further subdividing of these directories into SINUMERIK NCK / PLC / HMI and SINAMICS covers the data areas described above and is echoed in all named directories (data classes).

| CompactFlash card | Data area |
|---|---|
| SINUMERIK | NCK |
| | PLC |
| | HMI |
| SINAMICS | Drives |

## Structure of the CompactFlash card

The file system comprises the uppermost level of the directories "addon", "individual", "oem", "siemens" and "user": These directories have essentially an identical structure.

The section of the directory structure relevant for the configuration is shown below:

| siemens directory | | |
|---|---|---|
| /siemens/sinumerik | | |
| | /hmi | |
| | /appl | Applications (operating areas) |
| | /base | Basic system components |
| | /cfg | All configuration files |
| | /data | Version data |
| | /hlp | Online help files |
| | /hlps | Online help files, zipped and version files |

| **siemens directory** | | | |
|---|---|---|---|
| | /ico | | Icon files |
| | | /ico640 | Icons with a resolution 640x480 |
| | | /ico800 | Icons with a resolution 800x600 |
| | | /ico1024 | Icons with a resolution 1024x768 |
| | | /ico1280 | Icons with a resolution 1280x1024 |
| | | /ico1600 | Icons with a resolution 1600x1240 |
| | /lng | | Text files |
| | /lngs | | Text files zipped and versions files |
| | /osal | | |
| | | /ace | ACE/TAO |
| | | /qt | Qt |
| | /proj | | Easy Screen configurations |
| | /template | | Various templates |
| | | /cfg | Templates for configuration files |
| | | /ing | Templates for text files |
| | /tmpp | | Storage, temporary data |

| **addon directory** | | | |
|---|---|---|---|
| /addon/sinumerik | | | |
| | /hmi | | |
| | /appl | | Applications (operating areas) |
| | /cfg | | Configuration files |
| | /data | | Version data |
| | /hlp | | Online help files, zipped and version files |
| | /ico | | Icon files |
| | | /ico640 | Icons with a resolution 640x480 |
| | | /ico800 | Icons with a resolution 800x600 |
| | | /ico1024 | Icons with a resolution 1024x768 |
| | | /ico1280 | Icons with a resolution 1280x1024 |
| | | /ico1600 | Icons with a resolution 1600x1240 |
| | /lng | | Text files |
| | /lngs | | Text files zipped and versions files |
| | /proj | | Easy Screen configurations |
| | /template | | Various templates |

| **oem directory** | | | |
|---|---|---|---|
| /oem/sinumerik | | | |
| | /data | | Version data |
| | | /archive | Manufacturer archives |
| | /hmi | | |

| **oem directory** | | | |
|---|---|---|---|
| | /appl | | Applications (operating areas) |
| | /cfg | | Configuration files |
| | /data | | Version data |
| | /hlp | | Online help files |
| | /hlps | | Online help files, zipped and version files |
| | /ico | | Icon files |
| | | /ico640 | Icons with a resolution 640x480 |
| | | /ico800 | Icons with a resolution 800x600 |
| | | /ico1024 | Icons with a resolution 1024x768 |
| | | /ico1280 | Icons with a resolution 1280x1024 |
| | | /ico1600 | Icons with a resolution 1600x1240 |
| | /lng | | Text files |
| | /lngs | | Text files zipped and versions files |
| | /proj | | Easy Screen configurations |
| | /template | | Various templates |

| **user directory** | | | |
|---|---|---|---|
| /user/sinumerik | | | |
| | /data | | Version data |
| | | /archive | User-specific archives |
| | | /prog | User-specific programs |
| | /hmi | | |
| | | /cfg | Configuration files |
| | | /data | Version data |
| | | /hlp | Online help files |
| | | /ico | Icon files |
| | | | /ico640 | Icons with a resolution 640x480 |
| | | | /ico800 | Icons with a resolution 800x600 |
| | | | /ico1024 | Icons with a resolution 1024x768 |
| | | | /ico1280 | Icons with a resolution 1280x1024 |
| | | | /ico1600 | Icons with a resolution 1600x1240 |
| | | /lng | Text files |
| | | /log | Log files |
| | | /md | Machine data views |
| | | /proj | Easy Screen configurations |

# Operating software settings 3

## 3.1 Access levels

### Access to functions and machine data

The access concept controls access to functions and data areas. Access levels 0 to 7 are available, where 0 represents the highest level and 7 the lowest level. Access levels 0 to 3 are locked using a password and 4 to 7 using the appropriate key-operated switch.

| Access level | Locked by | Area | Data class |
|---|---|---|---|
| 1 | Password: SUNRISE | Manufacturer | Manufacturer (M) |
| 2 | Password: EVENING | Service | Individual (I) |
| 3 | Password: CUSTOMER | User | User (U) |
| 4 | Key-operated switch setting 3 | Programmer, machine setter | User (U) |
| 5 | Key-operated switch setting 2 | Qualified operator | User (U) |
| 6 | Key-operated switch setting 1 | Trained operator | User (U) |
| 7 | Key-operated switch setting 0 | Semi-skilled operator | User (U) |

The password remains valid until it is reset with the "Delete Password" softkey. The passwords can be changed after activation.

If, for example, the passwords are no longer known, reinitialization (boot-up with "NCK default data") must be carried out. This resets all passwords to the default (see table). POWER ON does not reset the password.

| NOTICE |
|---|
| **Changing the password preassignment**<br>The misuse of passwords can also represent a considerable security risk.<br>For simple commissioning, documented passwords are preassigned. We recommend that the passwords are regularly changed during productive operation. |

## Key-operated switch

Access levels 4 to 7 require a corresponding key-operated switch setting on the machine control panel. Three keys of different colors are provided for this purpose. Each of these keys provides access only to certain areas.

Meaning of the key-operated switch settings:

| Access level | Switch position | Key color |
|---|---|---|
| 4-7 | 0 to 3 | Red |
| 5-7 | 0 to 2 | Green |
| 6-7 | 0 and 1 | Black |
| 7 | 0 = Key removal position | No key inserted |

The key-operated switch setting must always be edited from the PLC user program and applied to the interface accordingly.

## References

The password can also be deleted via the → PI service: LOGOUT

Function Manual Basic Functions, Section "PLC for SINUMERIK 828D" (P4)

## 3.2 How to set and change the password

### Set password

To change the access level, select the "Start-up" operating area:

1. Press the "Password" softkey.
2. Press the "Set password" softkey to open the following dialog:

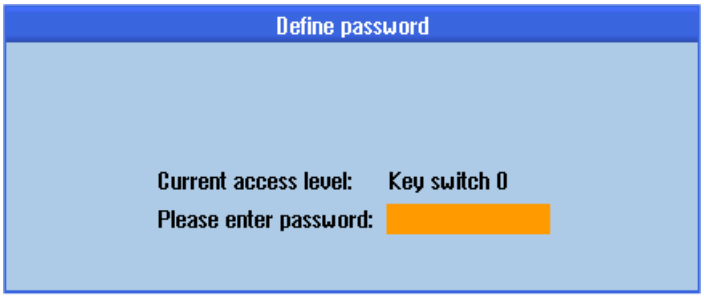



Figure 3-1 Set password

3. Enter a password and confirm this with "OK" or with the <Input> key.

   A valid password is acknowledged as set and the currently applicable access level is displayed. Invalid passwords will be rejected.

4. You must delete the old password before activating a password for a lower access level than the one activated.

   The last valid password is deleted by pressing the "Delete password" softkey. Then the current key-operated switch setting is valid.

### Change password

To change the password:

1. Press the "Change password" softkey to open the following dialog:

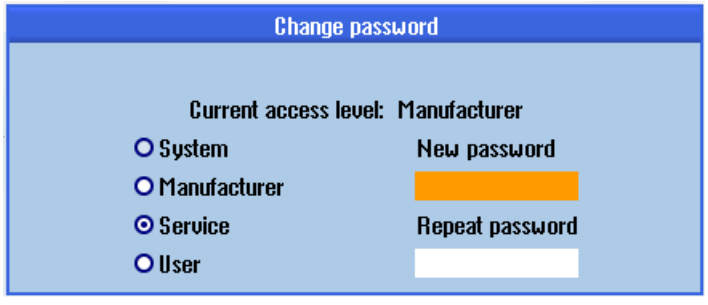



Figure 3-2 Change password

2. Enter the new password in both fields and then confirm with the "OK" softkey. If both passwords match, the new password becomes valid and is adopted by the system.

# 3.3 How to set the date and time

## Requirement

Changes can only be made with the appropriate access authorization (as of "User" and higher).

## Setting the date and time

Procedure:

1. Select the "Startup" operating area.
2. Press the "HMI" softkey.
3. Press the "Date/Time" softkey.

   The "Date/Time" window opens.
4. Select the required formats for the date and time in the "Format" field.
5. Confirm the entry with the "OK" softkey.

   The new date and time details are accepted and output on the first line in the "current" fields.

---

**Note**

**Synchronization of time**

The SINAMICS drives have no real-time clock. The time of day and date of the SINAMICS clock are synchronized in a 10-second cycle with the SINUMERIK real-time clock.

The result is that after a change of the date and/or the clock of the SINUMERIK real-time clock up to 10 seconds expire until this change has been synchronized with the SINAMICS drives.

If, in this period of up to 10 seconds, SINAMICS alarms occur (alarm numbers 200000 - 299999), these SINAMICS alarms still receive the outdated date / time stamp. The SINUMERIK alarms (alarm numbers < 200000 and > 300000) triggered as a result of the SINAMICS alarms, on the other hand, already receive the new date/time stamp.

---

# 3.4 Setting the operating software languages

## 3.4.1 Available system languages and language extensions

### System languages

In the basic configuration, the SINUMERIK 828D is delivered with the following system languages:

- English
- French
- German
- Italian
- Korean
- Portuguese (Brazil)
- Chinese (simplified)
- Spanish
- Chinese (traditional)

All system languages are installed in the SINUMERIK 828D as delivered, so that the language can be changed directly at the user interface, without having to download system language data.

### Additional languages

Additional languages are available on the language extensions DVD. Installation instructions as well as compatibility notes regarding the software version are provided on the DVD.

| NOTICE |
| --- |
| **Language extensions when upgrading**<br>After upgrading the system software, the system languages and the language extensions are available, which are compatible with the current software version. Language extensions that are no longer compatible are deleted. |

## 3.4.2 This is how you delete a language extension

### Requirement

Only language extensions, i.e. languages that have been subsequently installed, can be deleted again. System languages cannot be removed.

### Deleting a language

Procedure:

1. Select the "Startup" operating area and then the "Change language" softkey.
2. The "Language selection" window opens. The active language is highlighted.
3. Select the language that you wish to delete.
4. Press the "Delete language" softkey.
5. Confirm with "OK".

## 3.4.3 Entering Asian characters using the Input Method Editor

### Input Method Editor (IME)

The program editor and PLC alarm text editor both allow you to edit Asian characters. The Input Method Editor is available as support for the following Asian languages:

- Simplified Chinese
- Traditional Chinese (Taiwanese)
- Korean

The editor is started using the shortcut key <Alt>+<S>.

#### Editor with active learning function

For Chinese simplified and Chinese traditional, the system offers the option of working with dictionaries:

- Using a dictionary at the control
- Importing a dictionary into the control

The editor opens a learning function if a phonetic spelling is entered, for which there is no match in the control system. This function allows syllables or words to be combined, which are then permanently available after saving. The editor displays the combination of Chinese characters next to the pinyin phonetic spelling. After being completely compiled, the word should then be saved by pressing the <Input> key. It is simultaneously inserted in the associated text box.

## Editing the dictionary

When this function is activated, another line showing the combined characters and sounds will appear. The editor will then offer various characters for this sound, from which you can choose the desired one by entering the appropriate digit (1 ... 9). You can toggle the input cursor between the compound phonetic notations field and the phonetic input field by pressing the <TAB> key.




When the cursor is positioned in the upper field, you can undo the combination by pressing the <backspace> key.




Press <Select> to save the characters currently being displayed.




Press the <Delete> key if you want to delete the currently displayed group of characters from the dictionary.

## Importing a dictionary

A dictionary can now be generated using any Unicode editor by attaching the corresponding Chinese characters using the pinyin phonetic spelling. If the phonetic spelling contains several Chinese characters, then the line must not contain any additional match. If there are several matches for one phonetic spelling, then these must be specified in the dictionary line by line. Otherwise, several characters can be specified for each line.

The generated file should be saved in the UTF8 format under the name chs_user.txt (simplified Chinese) or cht_user.txt (traditional Chinese).

Line structure:

Pinyin phonetic spelling <TAB> Chinese character <LF>

OR

Pinyin phonetic spelling <TAB> Chinese character1<TAB> Chinese character2 <TAB> … <LF>

<TAB> - tab key

<LF> - line break

Store the created dictionary in one of the following paths:

```
../user/sinumerik/hmi/ime/
```

```
../oem/sinumerik/hmi/ime/
```

When the Chinese editor is called the next time, this enters the content of the dictionary into the system dictionary.

**Example:**

| | | | | | |
|---|---|---|---|---|---|
| ai | 哎 | 哀 | 唉 | 埃 | 挨 |
| caise | 彩色 | | | | |
| hongse | 紅色 | | | | |
| huise | 灰色 | | | | |
| heli | 河裏 | | | | |
| zuihaowan | 最好玩 | | | | |

**Note**

The dictionaries are also stored in the commissioning archive when "HMI data" is selected (see also: Which data is backed up in the archive? (Page 338))

## 3.4.4 This is how you enter the Chinese character

### Entering Chinese characters

A character is selected using a phonetic spelling ("pinyin method"): whose sound can be formed by combining Latin letters.

1. Enter the sound by combining Latin letters.

   The editor lists a selection of characters that correspond to that particular sound.

2. Select the desired character using the cursor keys:

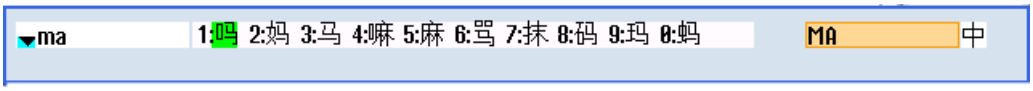



Figure 3-3 Example: Chinese simplified

If you switch over the option button to enter a Latin letters using the <SELECT> key, then the entries are directly transferred into the text box that had the focus before the Chinese editor was opened.

### Zhuin input for Chinese traditional

For traditional Chinese, the following input options are available:

1. Use the numerical block of the keyboard to form the individual syllables.

   Each number is assigned a certain number of letters, that can be selected by pressing the numeric key one or several times.

2. In order to accept the selection made, which is displayed in the Zhuyin text box, confirm the selection using the <INPUT> key or enter an additional number.

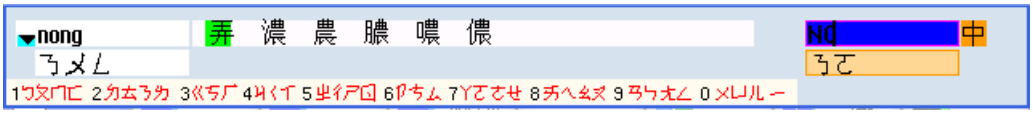



Figure 3-4 Example: Zhuin method

## 3.4.5 This is how you enter Korean characters

### Enter characters using a matrix

If only the keyboard is available at the control, then a matrix technique is used, which only requires the numerical block:

1. Using the first number, select the line: the line is then highlighted in color.
2. Using the second number, select the column: the character will be briefly highlighted in color and then transferred to the "Character" field.

You can toggle between Korean and Latin using the <SELECT> key.

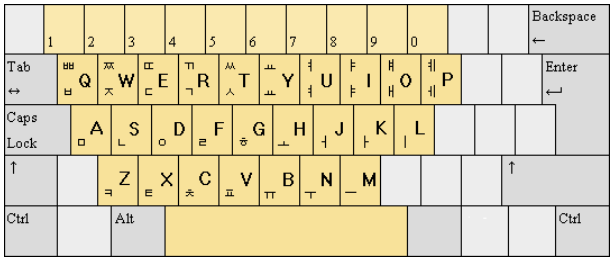



Figure 3-5 Example: Program editor

You can switch the input mode between "Matrix" and "Beolsik".

## Using the Korean keyboard

To enter Korean characters, you will need a keyboard with the keyboard assignment shown below. In terms of key layout, this keyboard is the equivalent of an English QWERTY keyboard and individual characters must be grouped together to form syllables.




Example: Korean keyboard

The Korean alphabet (Hangeul) consists of 24 letters: 14 consonants and 10 vowels. Syllables are formed by combining consonants and vowels.

# 3.5 Checking and entering licenses

## Options that require a license

The use of the installed system software and the options activated on a SINUMERIK control system require that the licenses purchased for this purpose are assigned to the hardware. In the course of this assignment, a license key is generated from the license numbers of the system software, the options, as well as the hardware serial number. Here, a license database administered by Siemens is accessed via the Internet. Finally, the license information including the license key is transferred to the hardware.

The license database can be accessed using the Web License Manager.

## Web License Manager

By using the Web License Manager, you can assign licenses to hardware in a standard Web browser. To conclude the assignment, the license key must be entered manually on the control via the user interface.

Internet address: Web License Manager (http://www.siemens.com/automation/license)

**Note**

**SINUMERIK software products**

If a license key has not been activated or does not exist for a SINUMERIK software product, alarm 8081 is output by the control and NC START cannot be executed.

## Activating the test license

In order to use an option without purchasing a license, you can activate the option for a limited time using the test license. You require as a minimum access level 2 "Service" to activate the option. The test license can be used for 150 hours, and can be used by the machine OEM for commissioning and testing.

## See also

Definitions for license management (Page 46)

## 3.5.1 How to enter a license key

## Precondition

The appropriate licenses are required for the activated options. After licensing the options in the Web License Manager, you receive a "license key" containing all options requiring a license and which is only valid for your system CompactFlash card.

To set or reset options the "Manufacturer" access level is required.

## Reading in or entering a license key

Procedure:

1. Select the "Start-up" operating area.
2. Press the menu forward key.
3. Press the "Licenses" softkey.
4. The "Licensing" window opens
5. Select the input line in order to enter a new license key.

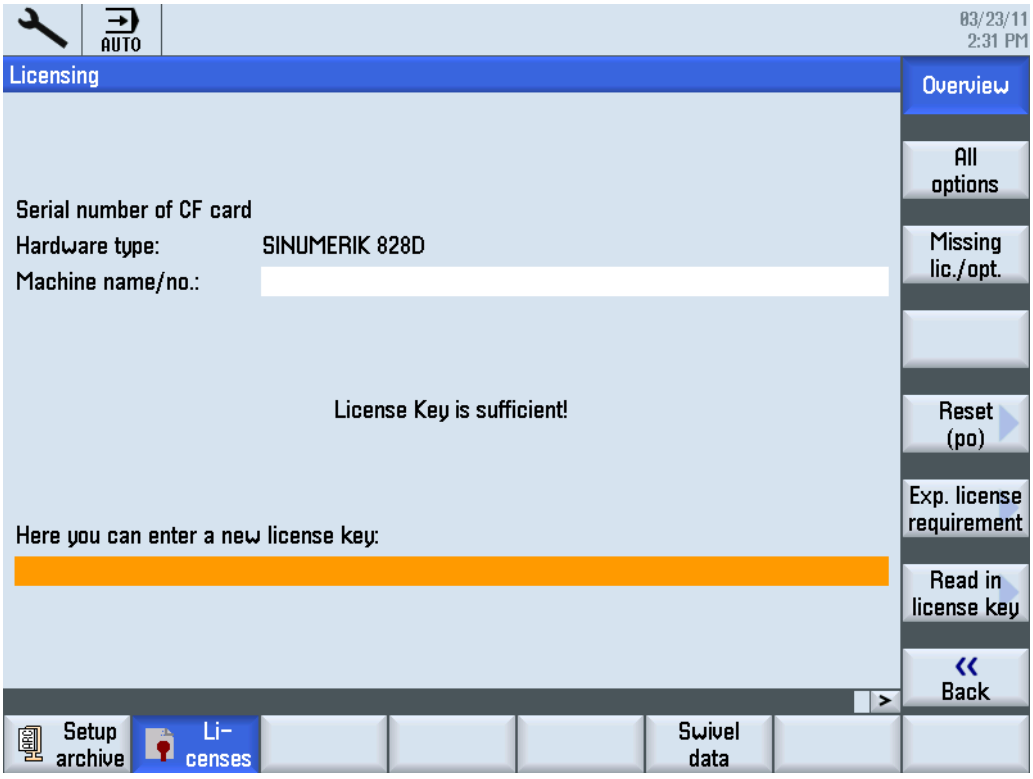



Figure 3-6 Licensing

Other actions include:

- Softkey "All options": All options are displayed.
- Softkey "Missing licenses/options": Missing licenses and options are displayed: This is how you determine missing licenses/options (Page 45)
- Softkey: "Export license requirement": The missing licenses are exported in a file and can be saved on a storage medium.
- Softkey: "Read in the license key": The license key is read in from the license file.

  This file is supplied or can be sourced from the Web License Manager (http://www.siemens.com/automation/license).

## 3.5.2 This is how you determine missing licenses/options

### Determining the license requirement

Procedure:

1. Press the "All options" softkey to list all the options that can be selected for this control.
2. Activate or deactivate the required options in the "Set" column:
   - Mark the checkbox
   - Enter the number of options
3. Press the "Missing lic./opt." softkey in order to display all options that are licensed. In the "Set" column, you can deselect the options that you do not require.

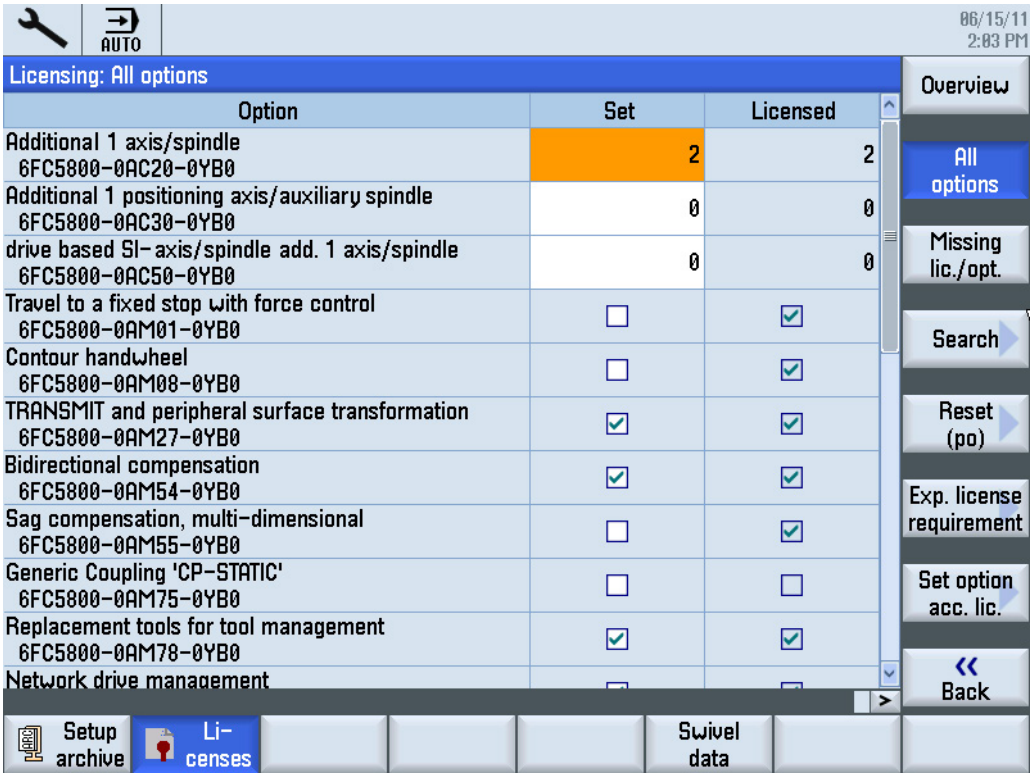



Figure 3-7 Licensing (example)

4. Press the softkey "Set option according to license", to activate all of the options contained in the license key. Confirm the following confirmation prompt with "OK".
5. To activate new selected options, press the "Reset (po)" softkey. A safety prompt appears.

1. Press the "OK" softkey to trigger a warm restart.

   - OR -

2. Press the "Cancel" softkey to cancel the process.

## 3.5.3 Definitions for license management

### Product

A product is marked by the data below within the license management of SINUMERIK → software products:

- Product label
- Article number
- → License number

### Software product

The term software product is generally used to describe a product that is installed on a piece of → hardware to process data. Within the license management of SINUMERIK software products, a corresponding → license is required to use each software product.

### Certificate of License (CoL)

The CoL is the proof of the → license. The product may only be used by the holder of the → license or authorized persons. The CoL includes the following data relevant for the license management:

- Product name
- → License number
- Delivery note number
- → Hardware serial number

### Hardware

In the context of license management of SINUMERIK → software products, hardware refers to the component of a SINUMERIK control system to which → licenses are assigned on the basis of its unique identifier. The license information is also saved to non-volatile memory on this component, e.g. on a → CompactFlash card.

### CompactFlash card

As the carrier of all non-volatile data in a SINUMERIK solution line control system, the CompactFlash card represents the identity of this control system. The CompactFlash card is a memory card that can be plugged into the → Control Unit from outside. The CompactFlash card also contains the following data relevant for the license management:

- → Hardware serial number
- License information including the → license key

### Hardware serial number

The hardware serial number is a permanent part of the → CompactFlash card. It is used to identify a control system uniquely. The hardware serial number can be determined by:

- → Certificate of License
- User interface
- Label on the CompactFlash card

### License

A license gives the user the legal right to use a → software product. Evidence of this right is provided by the following:

- → Certificate of License (CoL)
- → License key

### License number

The license number is the feature of a → license used for its unique identification.

### License key

The License Key is the "technical representative" of the sum of all the → licenses that are assigned to one particular piece of → hardware, which is uniquely marked by its → hardware serial number.

### Option

An option is a SINUMERIK → software product that is not contained in the basic version and which requires the purchase of a → license for its use.

# 3.6 Configuring PLC alarms

## 3.6.1 Structure of the PLC alarms

### Creating and loading the PLC alarms

The PLC alarms in the range from **700000 - 700247** are configured by the machine manufacturer. The access level "Manufacturer" is required with the appropriate password.

To enter the manufacturer PLC alarms via the user interface, select "HMI"→ "Alarm texts" in the "Startup" operating area. Then you receive the following selection:

| Alarm texts for | Name of the xml file |
|---|---|
| Manufacturer cycle alarms | oem_alarms_cycles |
| Manufacturer PLC alarms | oem_alarms_plc |
| Manufacturer part program message texts | oem_partprogram_messages |

The alarm text files are only loaded during startup.

- "Alarm" attribute: Red, is shown in the "alarm list".
- "Message" attribute: Black, is shown under "Messages".

To load the alarm texts, restart the system.

### Structure of a PLC alarm

The user PLC alarms have the following structure:

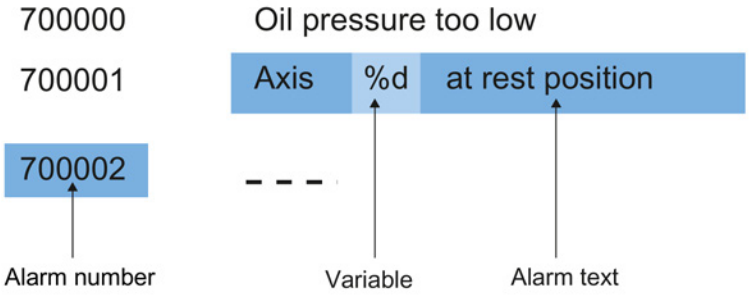



Figure 3-8 Alarm structure

The table below shows the mode of operation of the PLC alarms:

- The alarm is triggered with the appropriate number and output via the PLC signal.
- The alarm text can be freely selected and may be up to 255 characters long.
- The NC response when the alarm is triggered is defined in MD14516 $MN_USER_DATA_PLC_ALARM.

- For a control with more than one channel, the channel assignment is made via MD14518 $MN_USER_DATA_PLC_ALARM_ASSIGN.
- If a variable has been configured for an alarm, the value of this variable is in the specified data word of the PLC variable.

| Alarm number | PLC signal | PLC variable | Alarm response | Alarm text |
|---|---|---|---|---|
| 700000 | DB1600.DBX0.0 | DB1600.DBD1000 | MD14516[0] | Alarm 1 |
| 700001 | DB1600.DBX0.1 | DB1600.DBD1004 | MD14516[1] | Alarm 2 |
| 700002 | DB1600.DBX0.2 | DB1600.DBD1008 | MD14516[2] | Alarm 3 |
| 700003 | DB1600.DBX0.3 | DB1600.DBD1012 | MD14516[3] | Alarm 4 |
| 700004 | DB1600.DBX0.4 | DB1600.DBD1016 | MD14516[4] | Alarm 5 |
| 700005 | DB1600.DBX0.5 | DB1600.DBD1020 | MD14516[5] | Alarm 6 |
| 700006 | DB1600.DBX0.6 | DB1600.DBD1024 | MD14516[6] | Alarm 7 |
| ... | | | | |
| 700246 | DB1600.DBX30.6 | DB1600.DBD1987 | MD14516[246] | Alarm 247 |
| 700247 | DB1600.DBX30.7 | DB1600.DBD1988 | MD14516[247] | Alarm 248 |

## Configuring alarm texts with variables

The following data types are permitted for variables in the alarm text:

| Variable | Meaning |
|---|---|
| %b | Binary representation of a 32-bit value |
| %d | Integer decimal number |
| %f | 4-byte floating-point number |
| %i | Integer decimal number with sign |
| %o | Integer octal number |
| %u | Unsigned decimal number |
| %x | Integer hexadecimal number |

## References

You can find further details on the PLC alarms in the following manuals:

- SINUMERIK 840D sl/828D: Function Manual, Basic Functions: PLC for SINUMERIK 828D (P4), Section "PLC user alarms"
- SINUMERIK 828D Diagnostics Manual, Section "System responses"

## 3.6.2 How to create PLC alarms

### Notes for processing

The following points should be observed when processing the files:

- The files should be edited externally on a PG/PC with a text editor (e.g. notepad) or with an XML editor. The structure must not be altered.
- The created alarm text files are copied to the the following directory on the CompactFlash card: `oem/sinumerik/hmi/lng`
- To enable the system to recognize the alarm text file, the file name must be written in lower case letters.
- The alarm text file is converted during system startup: The system must be restarted to activate the alarms.

### Procedure

To edit a larger number of alarms, first create 2 or 3 alarms directly on the control. Then the file oem_alarms_plc_xxx.ts is created and you have a "document template" with the correct structure, which you can then extend with further alarms. The abbreviation "xxx" stands for the language in which the file has been created.

1. Select the "Start-up" operating area.
2. Press the "HMI" softkey.
3. Press the "Alarm texts" softkey. The "Select file" window appears.
4. Select "oem_alarms_plc" to create manufacturer PLC alarm texts.
5. Enter the alarm number in the "Number" field and the desired alarm text in the "Text" field. The alarm numbers and their alarm texts do not have to be consecutive. If an alarm is triggered without a configured text, only the alarm number is specified.

### Searching within the alarm texts

To search for a text or a series of characters:

1. Press the "Find >" softkey. The "Find" window opens and a new menu is displayed on the vertical softkey bar.
2. Enter the search term in the "Text" field.
3. Place the cursor in the "Direction" field and choose the search direction (forward, backward) with the "SELECT" key.
4. Activate the "Case-sensitive" checkbox when a distinction is to be made between upper and lower case in the entered text.
5. Press the "Find + replace" softkey. The "Find and replace" window appears.
6. Press the "OK" softkey to start the search.
7. Press the "Cancel" softkey to cancel the search.

Other navigation options are:

- Softkey "Go to start":

  The cursor jumps to the first entry of the selected alarm text file.

- Softkey "Go to end":

  The cursor jumps to the last entry of the selected alarm text file.

### See also

Table: Language code in file names (Page 349)

Example: How to create an online help for user PLC alarms (Page 65)

## 3.6.3 Configuring the alarm log

### Logging

Configure the alarm log in the "Diagnostics" operating area.

All alarms and messages are logged in chronological order with their raised and cleared time stamps. The exception are messages of the type "msg" from the NC part program. All alarms and messages that are no longer active when the log is displayed are also retained (historical alarm events).

The alarm log is organized as a ring buffer (default setting). The oldest entries are overwritten with new events in the following cases:

- When the maximum size is exceeded (permissible range: 0 - 32000).
- When the events happened before the last time the system was switched on.

### Permanent backup

Default setting: The alarm log is not backed up.

A backup of the alarm log is written to the `card/user/sinumerik/hmi/log/alarm_log` directory on the CompactFlash card. Set up a filter to limit the number of events in the alarm log and select a range:

- Time interval
- Alarm number range

| NOTICE |
| --- |
| **Service life of the CompactFlash card**<br>The service life of the CompactFlash card depends on the written data volume, not the number of write accesses.<br>With an average data volume of 60 MB per day, a service of more than ten years can be assumed. |

## 3.6.4 How to configure the log

### Configuring the log

Procedure:

1. Select the "Diagnostics" operating area.
2. Press the "Alarm log" softkey.
3. Press the "Settings" softkey.
4. Enter the desired number in the "Number of entries" field to change the maximum number of raised and cleared events.

   Default is 500 events; permissible value range 0 - 32000.
5. Select the type of logging under "File write mode":
   - "**Off**" if the events are not to be written to a file.
   - "**On every event**" if every event is to be written to a file.
   - "**Time controlled**" if the file is to be overwritten after a particular time interval. An additional "Time interval" input field appears in which you can specify the time in seconds.
6. Press the "Save log" softkey to save the alarm log.

The settings become effective only after restarting the system.

### Editing the configuration file

Procedure:

1. Copy the configuration file "oem_alarmprot_slaesvcconf.xml" from the `/siemens/sinumerik/hmi/template/cfg` directory.
2. Insert the file into the directory `/oem/sinumerik/hmi/cfg` or `/user/sinumerik/hmi/cfg`
3. Name the file "slaesvcconf.xml".

4. Open the user-specific file "slaesvcconf.xml" in the editor.
5. Enter the number of events to be output in the <Records type .../> identifier.

   The default value is 500. The permissible number is in the range from 0 ... 32000.

OR:

The number of events to be output and the type of logging can also be entered directly via the user interface:

1. Press the "Alarm log" → "Settings >" softkey in the "Diagnostics" operating area.

   As soon as changes are made to the default settings, the "slaesvcconf.xml" file is automatically created in the `/user/sinumerik/hmi/cfg` directory.
2. Enter the mode of the permanent storage in the <DiskCare type="int" value="-1"/> identifier. The following values are possible:

   **-1:** There is no saving of the alarm log (default setting).

   **0:** Each alarm event triggers an immediate saving of the alarm log.

   **>0:** Time for saving the log in seconds:

   When there is a change, the log is saved every n > 0 seconds.
3. You adapt the filter for the entry type in the <Filter> identifier.

   Here the following applies:

   – An alarm event is only entered in the log when it satisfies the filter criteria.
   – When several filters are defined, these should be linked using the logical operators OR or AND.

The settings become effective only after restarting the system.

---

**Note**

**Number of events**

Each incoming or outgoing event of an alarm or message requires a separate entry, even when they belong to the same alarm or message.

Acknowledgement events are also contained in the alarm log. They also require an entry even when they are not recognizable in the alarm log.

---

## Examples

All alarms that fulfill the following conditions are logged:

- CLEARINFO ≠ 15, therefore without part program messages:

```
<CONFIGURATION>
  <Protocol>
    <Filters>
      <Siemens_Filter_01 type="QString" value="CLEARINFO NOT 15" />
    </Filters>
  </Protocol>
</CONFIGURATION>
```

- "SEVERITY larger than 10" and "smaller than 500" :

```
<CONFIGURATION>
  <Protocol>
    <Filters>
      <Filter_01 type="QString" value= "SEVERITY HIGHER 10
 AND SEVERTY LOWER 500" />
    </Filters>
  </Protocol>
</CONFIGURATION>
```

# 3.7 Creating OEM-specific online help

## Overview

In addition to the existing system online help, you also have the option of creating a manufacturer-specific online help and adding this to the operator software.

This online help is generated in the HTML format, i.e. it comprises HTML documents that are linked with one another. The subject being searched for is called in a separate window from a contents or index directory. Similar to a document browser (e.g. Windows Explorer), a list of possible selections is displayed in the left-hand half of the window and when you click the required subject, the explanation is displayed in the right-hand half of the window.

Context-sensitive selection of online help pages is not possible.

General sequence:

1. Generating HTML files
2. Generating a help book
3. Integrating the online help in the operator software
4. Saving help files in the target system

## Other application cases

Online help for the following OEM-specific expansions can be generated and used to supplement the SINUMERIK Operate online help system:

- Online help for cycles and/or M functions of the machine manufacturer, which extend the programming options for SINUMERIK control systems. This online help is called in just the same way as the SINUMERIK Operate online help "Programming".

  **See also:** Example: This is how you generate a programming online help (Page 70)

- Online help for OEM-specific variables of the machine manufacturer. This online help is called from the variable view of SINUMERIK Operate.

  **See also:** Example: This is how you generate an online help for NC/PLC variables (Page 68)

## 3.7.1 Structure and syntax of the configuration file

### Syntax description of the "slhlp.xml"

You require the configuration file "slhlp.xml" to integrate the help book in the existing online help system of the user interface:

| Tag | Number | Meaning | |
|---|---|---|---|
| CONFIGURATION | 1 | Root element of the XML document: Indicates that this involves a configuration file. | |
| OnlineHelpFiles | 1 | Introduces the section of the help books. | |
| <help_book> | * | Introduces the section of a help book. | |
| EntriesFile | 1 | File name of the help book with the list of contents and subject (keyword) entries.<br>Attributes: | |
| | | value | Name of the XML file |
| | | type | Data type of the value (QString) |
| III-Technology | 0,1 | Specifies the technology for which the help book applies.<br>"All" applies for all technologies.<br>If the help book applies to several technologies, then the technologies are listed separated by comma.<br>Possible values:<br>All, Universal, Milling, Turning, Grinding, Stroking, Punching<br>Attributes: | |
| | | value | Technology data |
| | | type | Data type of the value (QString) |
| DisableSearch | 0,1 | Disable the subject (keyword) search for the help book.<br>Attributes: | |
| | | value | true, false |
| | | type | type, data type of the value (bool) |
| DisableFullTextSearch | 0,1 | Disable the full text search for the help book.<br>Attributes: | |
| | | value | true, false |
| | | type | type, data type of the value (bool) |
| DisableIndex | 0,1 | Disable the subject index for the help book.<br>Attributes: | |
| | | value | true, false |
| | | type | type, data type of the value (bool) |
| DisableContent | 0,1 | Disable the table of contents for the help book.<br>Attributes: | |
| | | value | true, false |
| | | type | type, data type of the value (bool) |

| Tag | Number | Meaning | |
|---|---|---|---|
| DefaultLanguage | 0,1 | Abbreviation for the language that should be displayed if the actual country language is available for the help book.<br>Attributes: | |
| | | value | chs, cht, deu, eng, esp, fra, ita, kor, ptb ... |
| | | type | Data type of the value (QString) |

The following applies to the "Number" column: * means 0 or several

### Example of a "slhlp.xml" file

The help book "hmi_myhelp.xml" is configured in the following example; the subject index is not activated:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE CONFIGURATION>
     <CONFIGURATION>
       <OnlineHelpFiles>
         <hmi_myhelp>
           <EntriesFile value="hmi_myhelp.xml" type="QString"/>
           <DisableIndex value="true" type="bool"/>
         </hmi_myhelp>
       </OnlineHelpFiles>
     </CONFIGURATION>
```

## 3.7.2 Structure and syntax of the help book

### Syntax for the help book

The help book is an XML file in which the structure of the online help is defined. The name of the file can be freely selected, e.g. "hmi_myhelp". In this file, you define:

- HTML documents
- Contents and subject index

| Tag | Number | Meaning |
|---|---|---|
| HMI_SL_HELP | 1 | Root element of the XML document |
| BOOK | | Identifies a help book. The name can be freely selected under the constraint that no name predefined by the system is used (such as sinumerik_alarm_plc_pmc).<br>Attributes: |

| Tag | Number | Meaning | |
|---|---|---|---|
| | + | ref | Identifies the HTML document that is displayed as the entry page for the help book. |
| | | titel | Title of the help book that is displayed in the table of contents. |
| | | helpdir | Directory that contains the online help of the help book. |
| ENTRY | * | Chapter of the online help<br>Attributes: | |
| | | ref | Identifies the HTML document that is displayed as entry page for the chapter. |
| | | titel | Title of the chapter that is displayed in the table of contents. |
| INDEX_ENTRY | * | Subject (keyword) to be displayed<br>Attributes: | |
| | | ref | Identifies the HTML document that is jumped to for this subject index entry. |
| | | titel | Title of the subject that is displayed in the subject index. |

The following applies to the "Number" column:

* means 0 or several

+ means 1 or several

## Formatting the index

You have the following options to format the subject index:

- Single entry: `<INDEX_ENTRY ...title="index"/>`
- Two two-stage entry, whereby each title has a main and a subentry.

  Separate the entries with a comma.

  ```
  <INDEX_ENTRY ...title="mainIndex_1,subIndex_1 with mainIndex_1"/>
  ```

- Two-stage entry, whereby the first title is the main entry and the second title is the subentry.

  Separate the entries with a semicolon.

  ```
  <INDEX_ENTRY ...title="mainIndex_2;subIndex_2
  without mainIndex_1"/>
  ```

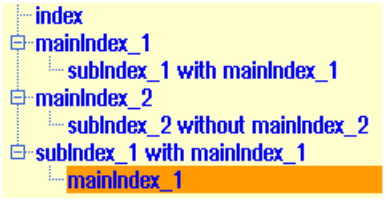



Figure 3-9 Example: Two-stage index

## 3.7.3 Description of the syntax for the online help

### Rules for creating the HTML files

Generating help files in the HTML format. It is possible to save all information in a single HTML file or to distribute the information over several HTML files.

The file names are assigned taking into account the following rules:

- References within HTML files should always be specified with relative paths. Only then can it be ensured that the references function in precisely the same way on both the development computer as well as on the target system.
- If jumps are to be made to certain points within an HTML file per link, then so-called anchors must be defined for this purpose.

  Example of an HTML anchor:

  ```
  <a name="myAnchor">This is an anchor</a>
  ```

  See also: Example: How to create an online help for user PLC alarms (Page 65)
- The contents of HTML documents must be saved with the UTF-8 coding. This guarantees that the HTML documents are correctly displayed in all of the supported country languages.

### HTML tags

The following sub-set of the HTML functional scope is supported:

| Tag | Description | Comment |
|---|---|---|
| a | Anchor or link | Supported attributes: href and name |
| address | Address | |
| b | Bold | |
| big | Larger font | |
| blockquote | Indented paragraph | |
| body | Document body | Supported attributes: bgcolor (#RRGGBB) |
| br | Line break | |
| center | Centered paragraph | |
| cite | Inline citation | Same effect as tag i |
| code | Code | Same effect as tag tt |
| dd | Definition data | |
| dfn | Definition | Same effect as tag i |
| div | Document division | The standard block attributes are supported |
| dl | Definition list | The standard block attributes are supported |
| dt | Definition term | The standard block attributes are supported |
| em | Emphasized | Same effect as tag i |
| font | Font size, family, color | Supported attributes: size, face, and color (#RRGGBB) |
| h1 | Level 1 heading | The standard block attributes are supported |
| h2 | Level 2 heading | The standard block attributes are supported |

| Tag | Description | Comment |
|---|---|---|
| h3 | Level 3 heading | The standard block attributes are supported |
| h4 | Level 4 heading | The standard block attributes are supported |
| h5 | Level 5 heading | The standard block attributes are supported |
| h6 | Level 6 heading | The standard block attributes are supported |
| head | Document header | |
| hr | Horizontal line | Supported attributes: width (can be specified as absolute or relative value) |
| html | HTML document | |
| i | Italic | |
| img | Image | Supported attributes: src, width, height |
| kbd | User-entered text | |
| meta | Meta-information | |
| li | List item | |
| nobr | Non-breakable text | |
| ol | Ordered list | The standard attributes for lists are supported |
| p | Paragraph | The standard block attributes are supported (default setting: left-aligned) |
| pre | Preformated text | |
| s | Strikethrough | |
| samp | Sample code | Same effect as tag tt |
| small | Small font | |
| span | Grouped elements | |
| strong | Strong | Same effect as tag b |
| sub | Subscript | |
| sup | Superscript | |
| table | Table | Supported attributes: border, bgcolor (#RRGGBB), cell-spacing, cellpadding, width (absolute or relative), height |
| tbody | Table body | No effect |
| td | Table data cell | The standard attributes for table cells are supported |
| tfoot | Table footer | No effect |
| th | Table header cell | The standard attributes for table cells are supported |
| thead | Table header | This is used to print tables that extend over several pages |
| title | Document title | |
| tr | Table row | Supported attributes: bgcolor (#RRGGBB) |
| tt | Typewrite font | |
| u | Underlined | |
| ul | Unordered list | The standard attributes for lists are supported |
| var | Variable | Same effect as tag tt |

### Block attributes

The following attributes are supported by the tags div, dl, dt, h1, h2, h3, h4, h5, h6, p:

- align (left, right, center, justify)
- dir (ltr, rtl)

### Standard attributes for lists

The following attributes are supported by tags ol and ul:

- type (1, a, A, square, disc, circle)

### Standard attributes for tables

The following attributes are supported by tags td and th:

- width (absolute, relative, no-value)
- bgcolor (#RRGGBB)
- colspan
- rowspan
- align (left, right, center, justify)
- valign (top, middle, bottom)

### CSS properties

The following table includes the supported CSS functional scope:

| Property | Values | Description |
|---|---|---|
| background-color | <color> | Background color for elements |
| background-image | <uri> | Background image for elements |
| color | <color> | Foreground color for text |
| text-indent | <length>px | Indent the first line of a paragraph in pixels |
| white-space | normal \| pre \| nowrap \| pre-wrap | Defines how whitespace characters are handled in HTML documents |
| margin-top | <length>px | Width of the upper edge of the paragraph in pixels |
| margin-bottom | <length>px | Width of the lower edge of the paragraph in pixels |
| margin-left | <length>px | Length of the left hand edge of the paragraph in pixels |
| margin-right | <length>px | Width of the righthand edge of the paragraph in pixels |
| vertical-align | baseline \| sub \| super \| mid-dle \| top \| bottom | Vertical alignment for text (in tables, only the values middle, top and bottom are supported) |
| border-color | <color> | Border color for text tables |

| Property | Values | Description |
|---|---|---|
| border-style | none \| dotted \| dashed \| dot-dash \| dot-dot-dash \| solid \| double \| groove \| ridge \| inset \| outset | Border style for text tables |
| background | [ <'background-color'> \|\| <'background-image'> ] | Short notation for background property |
| page-break-before | [ auto \| always ] | Page break before a paragraph/table |
| page-break-after | [ auto \| always ] | Page break after a paragraph/table |
| background-image | <uri> | Background image for elements |

### Supported CSS selectors

All CSS 2.1 selector classes are supported with the exception of so-called pseudo selector classes such as :first-child, :visited and :hover.

## 3.7.4 Example: This is how you generate an OEM-specific online help book

### Preconditions

Generate the following files:

- Configuration file: "slhlp.xml"

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE CONFIGURATION>
         <CONFIGURATION>
           <OnlineHelpFiles>
             <hmi_myhelp>
               <EntriesFile value="hmi_myhelp.xml" type="QString"/>
               <DisableIndex value="false" type="bool"/>
             </hmi_myhelp>
           </OnlineHelpFiles>
         </CONFIGURATION>
```

- Definition of the help book: "hmi_myhelp.xml"

```
<?xml version="1.0" encoding="UTF-8"?>
<HMI_SL_HELP language="en-US">
         <BOOK ref="index.html" title="Easy Help" helpdir="hmi_myhelp">
           <ENTRY ref="chapter_1.html" title="Chapter 1">
```

```
            <INDEX_ENTRY ref="chapter_1.html" title="Keyword 1"/>
            <INDEX_ENTRY ref="chapter_1.html" title="Keyword 2"/>
          </ENTRY>
          <ENTRY ref="chapter_2.html" title="Chapter 2">
            <INDEX_ENTRY ref="chapter_2.html" title="Keyword 2"/>
          </ENTRY>
          <ENTRY ref="chapter_3.html" title="Chapter 3">
            <INDEX_ENTRY ref="chapter_3.html" title="Keyword 3"/>
            <ENTRY ref="chapter_31.html" title="Chapter 3.1">
              <INDEX_ENTRY ref="chapter_31.html" title="test;Chapter 3.1"/>
            </ENTRY>
            <ENTRY ref="chapter_32.html" title="Chapter 3.2">
              <INDEX_ENTRY ref="chapter_32.html" title="test;Chapter 3.2"/>
            </ENTRY>
          </ENTRY>
        </BOOK>
</HMI_SL_HELP>
```

## Saving help files in the target system

In the following example, the structure of a help book with the name "Easy Help" with table of contents and subject index is described.

Procedure:

1. Copy the configuration file "slhlp.xml" to the following directory:

   `/oem/sinumerik/hmi/cfg`

2. Create a directory for the desired language of the online help under the following path: `/oem/sinumerik/hmi/hlp`

   Use the specified language code from Chapter Table: Language code in file names (Page 349).

   **Note**

   **Notation**

   The directory names must be written in lower case.

   For example, if you are integrating a help in English, create an "eng" folder.

3. Place the help book, e.g. "hmi_myhelp.xml" in the "eng" folder.

   `/oem/sinumerik/hmi/hlp/eng/hmi_myhelp.xml`

4. Copy the help files to the following directory:

   `/oem/sinumerik/hmi/hlp/eng/hmi_myhelp/`

The settings become effective only after restarting the system.

**Note**

**Updates or changes**

When displaying the table of contents and index of a help book, for quicker processing, the help files are saved in the binary format in the `/siemens/sinumerik/sys_cache/hmi/hlp` directory: slhlp_<Hilfebuch>_*_<lng>.hmi .

In the example: slhlp_hmi_myhelp_*_eng.hmi

These files must first be deleted so that the changes can take effect and be displayed in the online help.

## Result

The book consists of three chapters with sections:

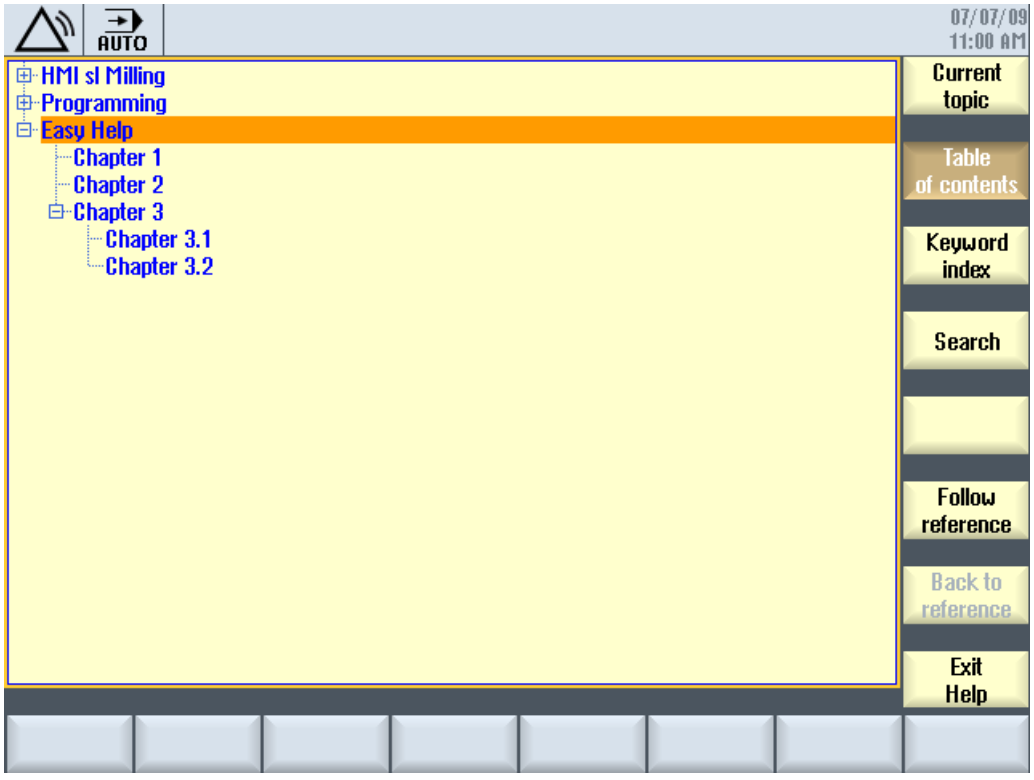



Figure 3-10 Example: OEM online help

Entries in the subject index:

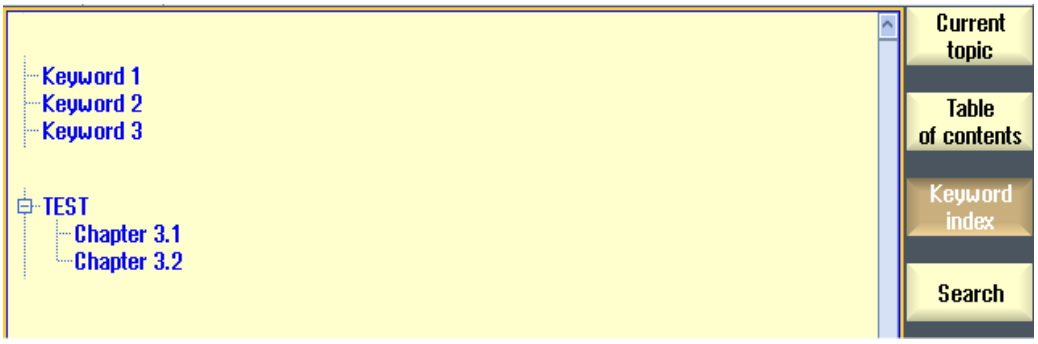



Figure 3-11 Example: Index

## 3.7.5 Example: How to create an online help for user PLC alarms

### Overview

If a user PLC alarm is triggered, a context-sensitive online help can be created for the respective alarm, e.g. with explanation and remedy. The online help texts for the user PLC alarms are managed in the following file: "sinumerik_alarm_oem_plc_pmc.html"

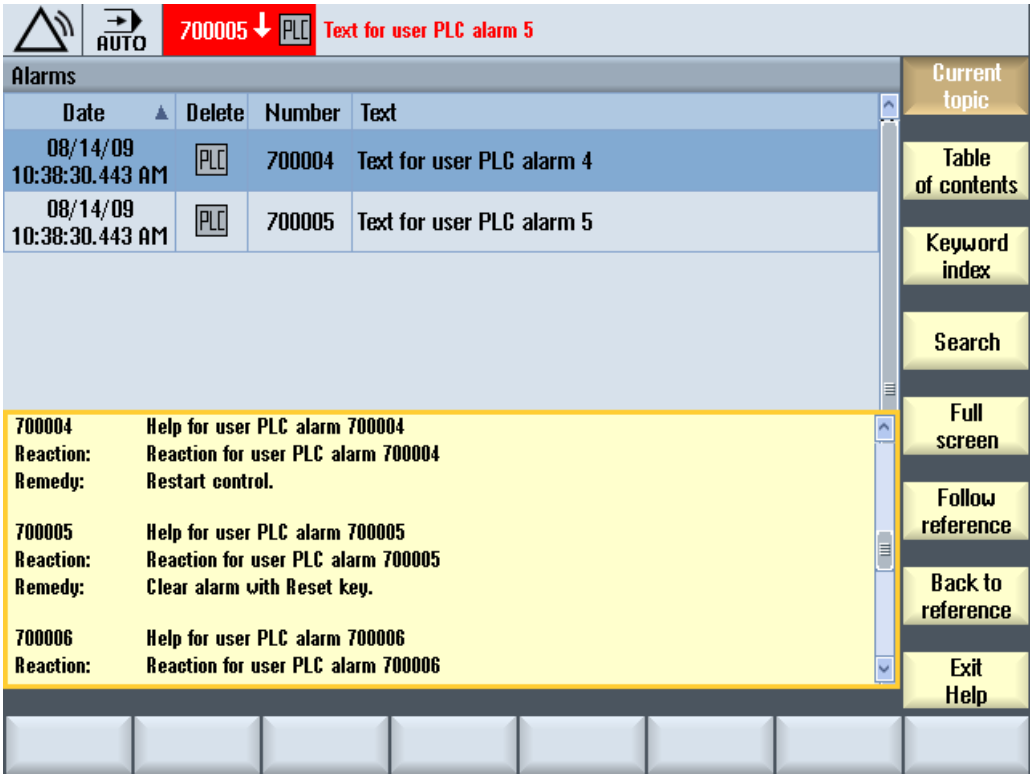



Figure 3-12 Example: Online help of user PLC alarms

## Structure of the help file

The following entries are permitted in the help file:

| Entry | Meaning |
|---|---|
| <a name="AlarmNo">AlarmNo</a> | Hyperlink to the alarm number |
|  ..... | Help text for the corresponding alarm |
| <td width="85%">.....</td> | Text that is displayed after the "Explanation" or "Remedy" field. |

## Creating a help file

The file name is **language-neutral** and must be:

**sinumerik_alarm_oem_plc_pmc.html**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE html PUBLIC>

<html>
<head><title></title></head>
<body>
<table>
...
<tr>
<td width="15%">
<b><a name="700004">700004</a></b></td>
<td width="85%"><b>Help for user PLC alarm 700004 </b></td></tr>
<tr><td valign="top" width="15%"><b>Reaction: </b></td>
<td width="85%">Reaction for user PLC alarm 700004 </td></tr>
<tr><td valign="top" width="15%"><b>Remedy:</b></td>
<td width="85%">Restart control. </td>
</tr>
<br>

<tr>
<td width="15%">
<b><a name="700005">700005</a></b></td>
<td width="85%"><b>Help for user PLC alarm 700005 </b></td></tr>
<tr><td valign="top" width="15%"><b>Reaction: </b></td>
<td width="85%">Reaction for user PLC alarm 700005 </td></tr>
<tr><td valign="top" width="15%"><b>Remedy:</b></td>
<td width="85%">Clear alarm with Reset key. </td>
</tr>
<br>

...
```

**sinumerik_alarm_oem_plc_pmc.html**

```
</table>
<p></p>
</body>
</html>
```

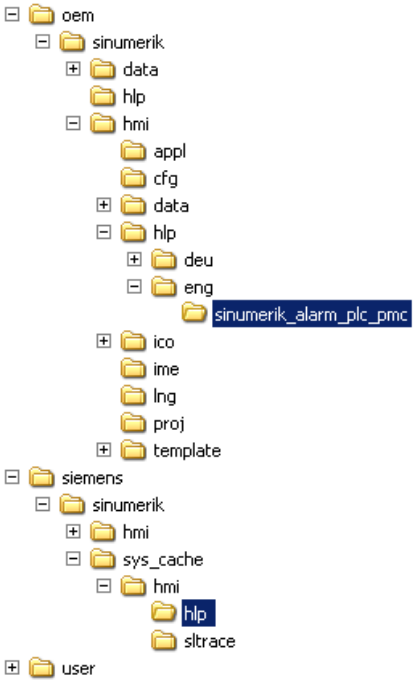
Procedure:

1. Copy the file to one of the following directories:

   `/oem/sinumerik/hmi/hlp/<lng>/sinumerik_alarm_plc_pmc/`

   `/user/sinumerik/hmi/hlp/<lng>/sinumerik_alarm_plc_pmc/`

   <lng> stands for the language code.




2. Delete all files in the directory:

   `/siemens/sinumerik/sys_cache/hmi//hlp`

The settings become effective only after restarting the system.

## 3.7.6 Example: This is how you generate an online help for NC/PLC variables

### Overview

In order to generate context-sensitive online help for NC/PLC variables or system variables as shown in the following example, the descriptive texts are managed in language-dependent html files.

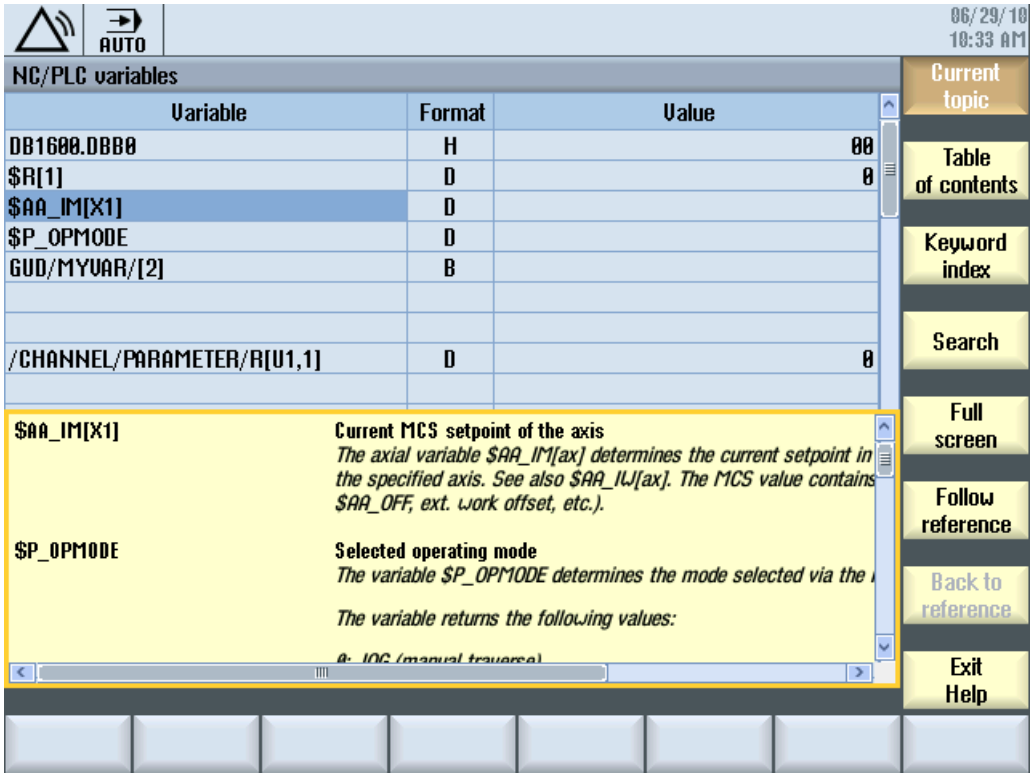



Figure 3-13 Example: Online help for user variables

### Structure of the online help

The following files are required for the online help:

| File | Meaning |
|---|---|
| sldgvarviewhelp.ini | Configuration file to manage an html file or several html files |
| <lng>/<name>1.html | The contents of all html files of the online help are language-dependent and are saved in the relevant language directory <lng>. |
| <lng>/<name>2.html | |
| . . . | |
| <lng>/<name>n.html | |

## Structure of the configuration file

The file is **language-neutral** and must be called:

**sldgvarviewhelp.ini**

```
[HelpBindings]
/BAG/STATE/OPMODE = var1_help.html#var1
$AA_IM[X1] = var1_help.html
$R[1] = var1_help.html#var2
/Channel/Parameter/R[u1,1] = var2_help.html#var2
DB1600.DBB0 = var2_help.html#var1
GUD/MyVar[2] = var2_help.html
```

### Note

The html files can be generated with any html editor. A definition is provided in the configuration file as to which html files belong to the online help.

The description can comprise one or several html files: For example, one html file for each variable or several identical variables in one file.

**Procedure:**

1. Copy the configuration file to the following directory:

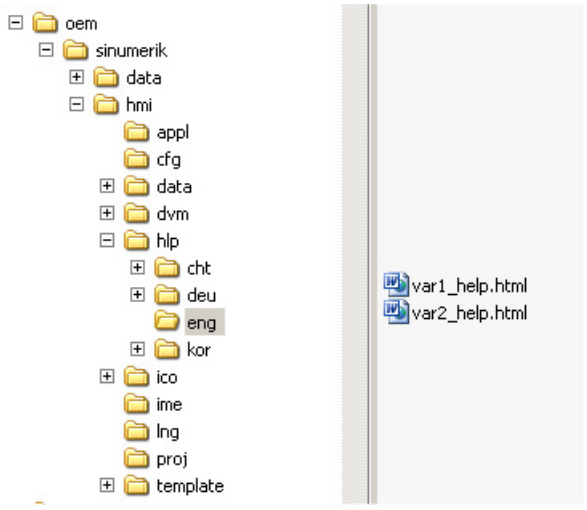
   `/oem/sinumerik/hmi/cfg/sldgvarviewhelp.ini`

2. Copy the html files to one of the following directories:

   `/oem/sinumerik/hmi/hlp/<lng>/`

   `/user/sinumerik/hmi/hlp/<lng>/`

   <lng> stands for the language code.




3. Delete all files in the directory:

   `/siemens/sinumerik/sys_cache/hmi//hlp`

The settings become effective only after restarting the system.

## 3.7.7 Example: This is how you generate a programming online help

### Requirements

Generate the following files:

- Configuration file "prog_help.ini"

```
[milling]
CYCLE1=cycle1_help.html
CYCLE2=cycle2_help.html#TextAnchor1
CYCLE3=cycle3_help.html
CYCLE4=cycle4_help.html
```

```
[turning]
CYCLE3=cycle2_help.html
CYCLE4=cycle3_help.html
```

- Configuration file for the help book "slhlp.xml" (optional)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE CONFIGURATION>
<CONFIGURATION>
  <OnlineHelpFiles>
    <hmi_prog_help>
      <EntriesFile value="hmi_prog_help.xml" type="QString"/>
      <DisableIndex value="true" type="bool"/>
    </hmi_prog_help>
  </OnlineHelpFiles>
</CONFIGURATION>
```

- Configuration file for the help book "hmi_prog_help.xml" (optional)

```
<?xml version="1.0" encoding="UTF-8"?>
<HMI_SL_HELP language="en-US">
  <BOOK ref="index.html" title="OEM_CYCLES" helpdir="hmi_prog_help">
    <ENTRY ref="cycle1_help.html" title="Cycle1"></ENTRY>
    <ENTRY ref="cycle2_help.html" title="Cycle2"></ENTRY>
    <ENTRY ref="cycle3_help.html" title="Cycle3"></ENTRY>
    <ENTRY ref="cycle4_help.html" title="Cycle4"></ENTRY>
    <ENTRY ref="cycle_help.html" title="OEM_Cycles"></ENTRY>
  </BOOK>
</HMI_SL_HELP>
```

- Language-dependent file "<prog_help_eng>.ts": this filename is permanently specified.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!DOCTYPE TS>
<TS>
     <context>
       <name>oem_cycles</name>
         <message>
```

```
            <source>CYCLE1</source>
            <translation>short description for cycle 1</translation>
            <chars>*</chars>
          </message>
          <message>
            <source>CYCLE2</source>
            <translation>short description for cycle 2</translation>
            <chars>*</chars>
          </message>
          <message>
            <source>CYCLE3</source>
            <translation>short description for cycle 3</translation>
            <chars>*</chars>
          </message>
      </context>
</TS>
```

## Saving help files in the target system

Procedure:

1. Copy the configuration file "prog.ini" to the following directory:

   `/oem/sinumerik/hmi/cfg`

2. Copy the file "`slhlp.xml`" to the help book in the following directory:

   `/oem/sinumerik/hmi/cfg`

3. Create a directory for the desired language of the online help under the following path: `/oem/sinumerik/hmi/hlp/<lng>` and copy the `hmi_prog_help.xml` file there

   Use the specified language code from the list of language codes for file names. The directory names must be written in lower case.

4. Copy the language-dependent `prog_help_<lng>.ts` file for the brief description to the following path: `/oem/sinumerik/hmi/<lng>/prog_help_<lng>.ts`

5. Copy the html files with the description of the OEM cycles to the following directory:

   `/oem/sinumerik/him/hlp/<lng>/hmi_prog_help/cycle<n>_help.html`

The settings become effective only after restarting the system.

## See also

Example: This is how you generate an OEM-specific online help book (Page 62)

# Parameterizing NC machine data

# 4

## 4.1 Classification of machine data

### Authorization for machine data

At least the Manufacturer password is necessary to enter or change machine data.

| ⚠WARNING |
| --- |
| **Lethal danger and danger for machines**<br>Changes in the machine data have a considerable influence on the machine.<br>Incorrect configuration of the parameters can endanger human life and cause damage to the machine. |

### Classification of machine data

The machine data is divided into the following areas:

- General machine data ($MN )
- Channel-specific machine data ($MC )
- Axis-specific machine data ($MA )
- Display machine data ($MM )
- General setting data ($SN )
- Channel-specific setting data ($SC )
- Axis-specific setting data ($SA )
- SINAMICS machine data (Control Unit and drive machine data):

  r0001 ... r9999 (read-only)

  r0001 ... r9999 (read/write)

**Note**

Machine data for turning and milling technologies is already set up in such a way that an adjustment of machine data is only necessary in exceptional cases.

A separate list image is provided for each of these areas in which you can view and edit machine data:

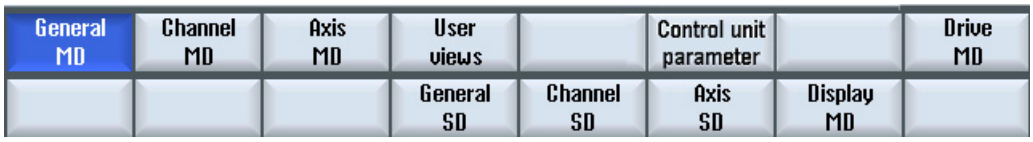



Figure 4-1 Softkey bar

The following properties of the machine data are displayed from left to right:

- Number of the machine data, if applicable with array index in square brackets
- Name of the machine data
- Value of the machine data
- Unit of the machine data
- Effectiveness of the machine data
- Data class

## Physical units of the machine data

The physical units of machine data are displayed on the right-hand side of the input field:

| Display | Unit | Measured variable |
|---|---|---|
| m/s**2 | m/s² | Acceleration |
| rev/s**3 | rev/s³ | Acceleration change for the rotating axis |
| kg/m**2 | kgm² | Moment of inertia |
| mH | mH (millihenry): | Inductance |
| Nm | Nm (Newton meters): | Torque |
| us | µs (microseconds): | Time |
| µA | µA (microamperes): | Electric current |
| µVs | µVs (microvolt-seconds): | Magnetic flux |
| userdef | user-defined: | The unit is defined by the user. |

If the machine data does not use units, no units are displayed.

If the data is not available, the "#" symbol is displayed instead of the value. If the value ends in an "H", it is a hexadecimal value.

## Effectiveness of the machine data

The right-hand column indicates when the changed machine data takes effect:

| | |
|---|---|
| cf | The changed machine data must be activated by pressing the "Apply MD" softkey. |
| po | The changed machine data requires an "NCK Power On-Reset" to take effect. |
| re | The changed machine data must be activated by pressing the <RESET> key. |
| so | The changed machine data becomes active immediately. |

## User views

User views are user-specific groups of machine data. They are used to call all relevant machine data in a certain operating state from various areas for processing.

The user views are stored on the CompactFlash card with the following path:

```
user/sinumerik/hmi/template/user_views
```

The following user views are already available as template:

- Electrical_Startup
- Mechanical_Startup
- Optimizing_Axis

## See also

Description of the data classes: Function Manual Basic Functions (P4)

A detailed description of the machine data is given in the list manual with cross-references to the appropriate section in the function manual.

# 4.2 Processing part programs from external CNC systems

**Note**

**Default setting**

For SINUMERIK 828D, the following machine data are already suitably preset for the particular technology – either turning or milling. These machine data cannot be changed and are not displayed.

## Activate the ISO Dialect function

The MD18800 $MN_EXTERN_LANGUAGE machine data is used to activate the external language. The ISO dialect M or T language types can be selected using MD10880 $MN_EXTERN_CNC_SYSTEM.

The two G commands of group 47 are used to switch from Siemens mode to ISO dialect mode:

- G290: Siemens NC programming language active
- G291: ISO dialect NC programming language active

The active tool, tool offsets and work offsets are retained. G290 and G291 must be the only components of an NC program block.

The switchover to an external programming language is contained in the scope of delivery for SINUMERIK 828D. MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB is only valid for NC language commands in the Siemens mode.

## References

Function Manual ISO Dialects

# 4.3 Machining free form surfaces using Advanced Surface

## Machining with Advanced Surface

When executing CAM programs in the high-speed cutting (HSC) range, the control has to process high feedrates with the shortest NC blocks. The result should be a good surface quality of the workpiece with high precision in the µm range at extremely high machining feedrates of > 10 m/min.

Using CYCLE832, the NC program is very precisely harmonized across the various machining strategies:

- When roughing, using smoothing of the contour, the emphasis is on speed.
- When finishing, the emphasis is on accuracy.

In both cases, specifying a tolerance ensures that the machining contour is maintained in order to achieve the desired surface quality.

## Checking the settings for Advanced Surface

After installation of the 828D Toolbox, several check programs of the MDC3AXV ... .SPF type are contained in the \Examples\AS_checkprograms directory.

The readme.txt file also contains further information in order to test the recommended machine data settings for Advanced Surface with the programs. When the program is started, a MDADVS.SPF file will be created in the SPF.DIR directory in which the test results are output as a list.

**Example:**

(extract from the complete file)

```
Machine Data setting different to required Advanced Surface Setting
Machine Data setting different to recommended Advanced Surface Setting
N20470 $MC_CPREC_WITH_FFW=1, recommended: 3
N20485 $MC_COMPRESS_SMOOTH_FACTOR[2]=0, recommended: 0.0001
N20485 $MC_COMPRESS_SMOOTH_FACTOR[3]=0, recommended: 0.0001
N20485 $MC_COMPRESS_SMOOTH_FACTOR[4]=0, recommended: 0.0001
N20486 $MC_COMPRESS_SPLINE_DEGREE[2]=3, recommended: 5
N20486 $MC_COMPRESS_SPLINE_DEGREE[3]=3, recommended: 5
N20486 $MC_COMPRESS_SPLINE_DEGREE[4]=3, recommended: 5
N20487 $MC_COMPRESS_SMOOTH_FACTOR_2[2]=0, recommended: 0.5
N20487 $MC_COMPRESS_SMOOTH_FACTOR_2[3]=0, recommended: 0.5
N20487 $MC_COMPRESS_SMOOTH_FACTOR_2[4]=0, recommended: 0.5
```

# 4.4 Machine data for Advanced Surface

## Machine data for Advanced Surface

The following setpoints are recommended for the listed machine data in order to achieve the optimum workpiece surface when machining mold workpieces:

**General machine data:**

| MD number | Designation | Default setting | Recom-mendation | Data class | Parameter |
|---|---|---|---|---|---|
| 10200 | $MN_INT_INCR_PER_MM | 100000 | 100000 | M | -- |
| 10210 | $MN_INT_INCR_PER_DEG | 100000 | 100000 | M | -- |
| 10682 | $MN_CONTOUR_SAMPLING_FACTOR | 1 | 1 | M | -- |

**Channel-specific machine data:**

| MD number | Designation | Default setting | Recom-mendation | Data class | Parameter |
|---|---|---|---|---|---|
| 20150[3] | $MC_GCODE_RESET_VALUES | 3 | 3 | M | FIFOCTRL |
| 20150[44] | $MC_GCODE_RESET_VALUES | 2 | 2 | M | UPATH |
| 20170 | $MC_COMPRESS_BLOCK_PATH_LIMIT | 20 | 20 | M | -- |
| 20172 | $MC_COMPRESS_VELO_TOL | 60000 | 1000 | M | -- |
| 20443[0] | $MC_LOOKAH_FFORM | 0 | 0 | M | DYNNORM |
| 20443[1] | $MC_LOOKAH_FFORM | 0 | 0 | M | DYNPOS |
| 20443[2] | $MC_LOOKAH_FFORM | 0 | 1 | M | DYNROUGH |
| 20443[3] | $MC_LOOKAH_FFORM | 0 | 1 | M | DYNSEMIFIN |
| 20443[4] | $MC_LOOKAH_FFORM | 0 | 1 | M | DYNFINISH |
| 20455 | $MC_LOOKAH_FUNCTION_MASK | 0 | 0 | M | -- |
| 20480 | $MC_SMOOTHING_MODE | 0 | 0 | M | -- |
| 20482 | $MC_COMPRESSOR_MODE | 100 | 100 | M | -- |
| 20485 | $MC_COMPRESS_SMOOTH_FACTOR[0-4] | 0 | 0 | M | -- |
| 20486 | $MC_COMPRESS_SPLINE_DEGREE[0-4] | 3 | 3 | M | -- |
| 20490 | $MC_IGNORE_OVL_FACTOR_FOR_ADIS | 1 | 1 | M | -- |
| 20550 | $MC_EXACT_POS_MODE | 3 | 3 | M | -- |
| 20600[0] | $MC_MAX_PATH_JERK | 10000 | 10000 | M | DYNNORM |
| 20600[1] | $MC_MAX_PATH_JERK | 10000 | 10000 | M | DYNPOS |
| 20600[2] | $MC_MAX_PATH_JERK | 10000 | 10000 | M | DYNROUGH |
| 20600[3] | $MC_MAX_PATH_JERK | 10000 | 10000 | M | DYNSEMIFIN |
| 20600[4] | $MC_MAX_PATH_JERK | 10000 | 10000 | M | DYNFINISH |
| 20602[0] | $MC_CURV_EFFECT_ON_PATH_ACCEL | 0 | 0 | M | -- |
| 20602[1] | $MC_CURV_EFFECT_ON_PATH_ACCEL | 0 | 0 | M | -- |
| 20602[2] | $MC_CURV_EFFECT_ON_PATH_ACCEL | 0,6 | 0,65 | M | -- |
| 20602[3] | $MC_CURV_EFFECT_ON_PATH_ACCEL | 0,6 | 0,6 | M | -- |

| MD number | Designation | Default setting | Recom-mendation | Data class | Parameter |
|---|---|---|---|---|---|
| 20602[4] | $MC_CURV_EFFECT_ON_PATH_ACCEL | 0,6 | 0,5 | M | -- |
| 20603[0] | $MC_CURV_EFFECT_ON_PATH_JERK | 0 | 0 | M | -- |
| 20603[1] | $MC_CURV_EFFECT_ON_PATH_JERK | 0 | 0 | M | -- |
| 20603[2] | $MC_CURV_EFFECT_ON_PATH_JERK | 0 | 0 | M | -- |
| 20603[3] | $MC_CURV_EFFECT_ON_PATH_JERK | 0 | 0 | M | -- |
| 20603[4] | $MC_CURV_EFFECT_ON_PATH_JERK | 0 | 0 | M | -- |
| 20605[0] | $MC_PREPDYN_SMOOTHING_FACTOR | 0 | 1 | M | -- |
| 20605[1] | $MC_PREPDYN_SMOOTHING_FACTOR | 0 | 1 | M | -- |
| 20605[2] | $MC_PREPDYN_SMOOTHING_FACTOR | 0 | 1 | M | -- |
| 20605[3] | $MC_PREPDYN_SMOOTHING_FACTOR | 0 | 1 | M | -- |
| 20605[4] | $MC_PREPDYN_SMOOTHING_FACTOR | 0 | 1 | M | -- |
| 20606[0] | $MC_PREPDYN_SMOOTHING_ON | 0 | 0 | M | DYNNORM |
| 20606[1] | $MC_PREPDYN_SMOOTHING_ON | 0 | 0 | M | DYNPOS |
| 20606[2] | $MC_PREPDYN_SMOOTHING_ON | 0 | 1 | M | DYNROUGH |
| 20606[3] | $MC_PREPDYN_SMOOTHING_ON | 0 | 1 | M | DYNSEMIFIN |
| 20606[4] | $MC_PREPDYN_SMOOTHING_ON | 0 | 1 | M | DYNFINISH |
| 28060 | $MC_MM_IPO_BUFFER_SIZE | 150 | 150 | M | -- |
| 28070 | $MC_MM_NUM_BLOCKS_IN_PREP | 80 | 80 | M | -- |
| 28302[0] | $MC_MM_PROTOC_NUM_ETP_STD_TYP | 28 | 28 | M | -- |
| 28520 | $MC_MM_MAX_AXISPOLY_PER_BLOCK | 3 | 5 | M | -- |
| 28530 | $MC_MM_PATH_VELO_SEGMENTS | 5 | 5 | M | -- |
| 28533 | $MC_MM_LOOKAH_FFORM_UNITS | 18 | 18 | M | -- |
| 28540 | $MC_MM_ARCLENGTH_SEGMENTS | 10 | 10 | M | -- |

**Axis machine data:**

| MD number | Designation | Default setting | Recom-mendation | Data class | Parameter |
|---|---|---|---|---|---|
| 32300 | $MA_MAX_ACCEL[0-3] | 1 | 1 | M | -- |
| 32300 | $MA_MAX_ACCEL[4] | 1 | 1 | M | -- |
| 32310[0] | $MA_MAX_ACCEL_OVL_FACTOR | 1.2 | 1.2 | M | -- |
| 32310[1] | $MA_MAX_ACCEL_OVL_FACTOR | 1.2 | 1.2 | M | -- |
| 32310[2] | $MA_MAX_ACCEL_OVL_FACTOR | 1.2 | 1.2 | M | -- |
| 32310[3] | $MA_MAX_ACCEL_OVL_FACTOR | 1.2 | 1.2 | M | -- |
| 32310[4] | $MA_MAX_ACCEL_OVL_FACTOR | 1.2 | 1.2 | M | -- |
| 32400 | $MA_AX_JERK_ENABLE | 0 | 0 | M | -- |
| 32402 | $MA_AX_JERK_MODE | 2 | 2 | M | -- |
| 32410 | $MA_AX_JERK_TIME | 0,001 | 0,001 | M | -- |
| 32431[0] | $MA_MAX_AX_JERK | 1000000 | 1000000 | I | DYNNORM |
| 32431[1] | $MA_MAX_AX_JERK | 1000000 | 1000000 | I | DYNPOS |
| 32431[2] | $MA_MAX_AX_JERK | 1000000 | 1000000 | I | DYNROUGH |

| MD number | Designation | Default setting | Recom-mendation | Data class | Parameter |
|---|---|---|---|---|---|
| 32431[3] | $MA_MAX_AX_JERK | 20 | 20 | I | DYNSEMIFIN |
| 32431[4] | $MA_MAX_AX_JERK | 20 | 20 | I | DYNFINISH |
| 32432[0] | $MA_PATH_TRANS_JERK_LIM | 1000000 | 1000000 | I | DYNNORM |
| 32432[1] | $MA_PATH_TRANS_JERK_LIM | 1000000 | 1000000 | I | DYNPOS |
| 32432[2] | $MA_PATH_TRANS_JERK_LIM | 1000000 | 1000000 | I | DYNROUGH |
| 32432[3] | $MA_PATH_TRANS_JERK_LIM | 20 | 20 | I | DYNSEMIFIN |
| 32432[4] | $MA_PATH_TRANS_JERK_LIM | 20 | 20 | I | DYNFINISH |
| 32620 | $MA_FFW_MODE | 3 | 3 | M | DYNNORM |
| 32640 | $MA_STIFFNESS_CONTROL_ENABLE | 1 | 1 | M | DSC |
| 33120 | $MA_PATH_TRANS_POS_TOL | 0,005 | 0,005 | U | -- |

**Setting data:**

| MD number | Designation | Default setting | Recom-mendation | Data class | Parameter |
|---|---|---|---|---|---|
| 42470 | $SC_CRIT_SPLINE_ANGLE | 36° | 36° | U | -- |
| 42471 | $SC_MIN_CURV_RADIUS | 1 | 1 | M | -- |
| 42500 | $SC_SD_MAX_PATH_ACCEL | 10000 | 10000 | U | -- |
| 42502 | $SC_IS_SD_MAX_PATH_ACCEL | 0 | 0 | U | -- |
| 42510 | $SC_SD_MAX_PATH_JERK | 10000 | 10000 | U | -- |
| 42512 | $SC_IS_SD_MAX_PATH_JERK | 0 | 0 | U | -- |

# 4.5 Machine data for an analog spindle

## Connection for encoder actual values

Precondition for connecting a spindle actual value encoder is an axis with digital drive (SINAMICS axis), which only has one encoder. A Sensor Module, e.g. SMC30, is connected as 2nd encoder at the axis module, which is connected with the directly mounted spindle actual value encoder. The PROFIdrive telegram of the SINAMICS axis must be designed for two encoder actual values. Depending on MD11240 $MN_PROFIBUS_SDB_NUMBER, the standard SDB of SINUMERIK 828D supports Siemens telegram 116 or Siemens telegram 136.

**Functional restrictions:**

- There is no setpoint telegram for the analog spindle; this means that the functionality based on the setpoint telegram is not available, e.g. DSC and torque feedforward control.
- The driver software simulates the "Drive ready" signal as being set.
- As drive parameters cannot be accessed, the associated functionality is restricted (e.g. spindle utilization display, automatic speed and position controller tuning). The missing values are shown in gray in the "Service Overview" dialog in the analog spindle column: "This axis is not affected."

## Connection for spindle setpoints

The analog spindle is connected to interface X252. Depending on the operating mode of the analog spindle, the following signals are output:

| Spindle type | Signal name | Meaning |
|---|---|---|
| Bipolar spindle | AOUT | Analog output +/-10 V |
| | AGND | Analog output 0 V reference signal |
| | DOUT11 | Controller enable |
| Unipolar spindle with separate enable and direction signals | AOUT | Analog output +10 V |
| | AGND | Analog output 0 V reference signal |
| | DOUT12 | Negative direction of travel (remains set even if there is no controller enable) |
| Unipolar spindle with direction-dependent enable | AOUT | Analog output +10 V |
| | AGND | Analog output 0 V reference signal |
| | DOUT11 | Controller enable and positive direction of travel |
| | DOUT12 | Controller enable and negative direction of travel |

The D/A conversion of the spindle setpoint is realized with a 14-bit resolution.

## Relevant machine data

The following machine data must be set for the analog spindle:

- Activation of the analog spindle:

  **MD30100 $MA_CTRLOUT_SEGMENT_NR = 0**

  The analog spindle for the setpoint output is activated by setting the local bus as bus segment.

- Selecting the analog spindle mode:

| MD30134 $MA_IS_UNIPOLAR_OUTPUT | |
|---|---|
| =0 | bipolar output (+/-10 V) |
| =1 | unipolar spindle with separate enable and direction signals |
| =2 | Unipolar spindle with direction-dependent enable |

- The rated output voltage must be adapted to the rated speed of the analog drive:

  **MD32250 $MA_RATED_OUTVAL**

  **MD32260 $MA_RATED_VELO**

- If the analog spindle is to be operated without an actual value encoder, then the value 0 must be entered as number of encoders:

  **MD30200 $MA_NUM_ENCS = 0**

- When using a directly mounted spindle actual value encoder, PROFIBUS should be selected as the actual value sensing type:

  **MD30240 $MA_ENC_TYPE = 1 or 4**

- The spindle actual value encoder must then be configured as 2nd encoder of a SINAMICS axis. To do this, the drive assignment of the actual value should be set the same as for the axis, where the SMC30 is to be connected at its axis module:

  **MD30220[0] $MA_ENC_MODULE_NR[0] = MD30220[0] of the SINAMICS axis**

- The input assignment of the actual value should be set to the input for the 2nd encoder at the SINAMICS axis module:

  **MD30230[0] $MA_ENC_INPUT_NR[0] = 2**

- Automatic drift compensation can be activated for a connected spindle actual value encoder:

  **MD36700 $MA_DRIFT_ENABLE**

  **MD36710 $MA_DRIFT_LIMIT**

- A basic drift value is continuously entered as additional speed setpoint independent of whether there is a spindle actual value encoder:

  **MD36720 $MA_DRIFT_VALUE**

## Example

Example of three axes and an analog spindle with actual value encoder (SMC30 as 2nd encoder connected to the Y axis).

| MD identifier | X | Y | Z | SP | Meaning |
|---|---|---|---|---|---|
| MD30100 CTRLOUT_SEGMENT_NR | 5 | 5 | 5 | 0 | Bus segment |
| MD30110 CTRLOUT_MODULE_NR | 1 | 2 | 3 | 1 | Module assignment |
| MD30120 CTRLOUT_NR | 1 | 1 | 1 | 1 | Output assignment |
| MD30130 CTRLOUT_TYPE | 1 | 1 | 1 | 1 | Output type |
| MD30134 IS_UNIPOLAR_OUTPUT | 0 | 0 | 0 | 0 | "0" means bipolar |
| MD30200 NUM_ENCS | 1 | 1 | 1 | 1 | Number of encoders |
| MD30220[0] ENC_MODULE_NR[0] | 1 | 2 | 3 | 2 | SMC30 on the Y axis |
| MD30230[0] ENC_INPUT_NR[0] | 1 | 1 | 1 | 2 | Input for second encoder |
| MD30240 ENC_TYPE | 1/4 | 1/4 | 1/4 | 1/4 | Encoder type |

| MD identifier | SP | Meaning |
|---|---|---|
| MD32250 RATED_OUTVAL | 80 (80% of 10 V) | Rated output voltage |
| MD32260 RATED_VELO | 3000 (at 8 V) | Rated motor speed |
| MD36700 DRIFT_ENABLE | 0 | Automatic drift compensation is deactivated. |
| MD36710 DRIFT_LIMIT | 0 | Limit for automatic drift compensation |
| MD36720 DRIFT_VALUE | 0 | Drift basic value |

## References

Additional information can be found in: Function Manual, Basic Functions, Spindles (S1)

# 4.6 Locking the control (CNC lock function)

## Preconditions

The following preconditions must be satisfied:

- Option requiring a license to be able to use the lock function:

| | |
|---|---|
|  | **Software option**<br>You require the following software option in order to use this function: CNC lock function |

- Access MyMachine, to activate the lock function.
- Programming tool to secure the lock function via PLC.

## Field of application

The OEM (machine manufacturer) decides to sell a machine using the business model with CNC Lock Function and licenses this function for the respective control system from Siemens using the serial number of the CompactFlash card. This means that the machine manufacturer has the option, using Access MyMachine, to save a date in the control system from which the NC start is locked. The CNC lock function is secured via the PLC.

| NOTICE |
|---|
| **Data required to release the system is lost**<br>In the case that the machine manufacturer loses the data required to release the control system, the control system as well as the CompactFlash card must be replaced. |

## General sequence

When commissioning the machine, the function must be activated in the PLC project and loaded to the control system together with the project.

The following procedure is required to activate the function:

- Access MyMachine: Create the file used for release.
- On the control system:
  - Correctly set date and time.
  - Import the file and restart the operating software.
  - Creating the commissioning archive.
- Access MyMachine: Create and save the file used for release.

The following procedure is required to deactivate the function:

- Import the file to release the control system and restart the operating software.

You can set a prewarning time with MD17300 $MN_CNC_LOCK_WARNING_TIME (default setting: 30 days). An alarm is output on a daily basis from this time onwards.

## References

Further details on this function are described in:

- Basic Functions Function Manual, Section "PLC for SINUMERIK 828D" (P4)
- Online help for SINUMERIK Integrate Access MyMachine /P2P

# 4.7 Managing data

## Application

The "Manage data" function is used to support and simplify the commissioning and provides functions for backing up, loading and comparing the following data:

- Machine data
- Setting data
- Drive data
- Compensation data

In contrast to a commissioning archive, only a single control object (axis, channel, servo, infeed, etc.) is saved in ASCII format (*.tea). This file can be edited and transferred to other control objects of the same type. The "Manage data" function is also used to transfer drive objects for SINAMICS drives.

## Managing data

The "Manage data" function offers the following options:

- Transfer data within the control
- Save data to a file
- Load data in a file
- Compare data

The following data can be saved and stored under the following absolute path on the CompactFlash card:

- **user**/sinumerik/hmi/data/backup/ec for compensation data
- **user**/sinumerik/hmi/data/backup/md for machine data
- **user**/sinumerik/hmi/data/backup/sd for setting data
- **user**/sinumerik/hmi/data/backup/snx for SINAMICS parameters

The function is called from "Commissioning" → "Machine data" → "Manage data".

### Example of "transferring data within the control"

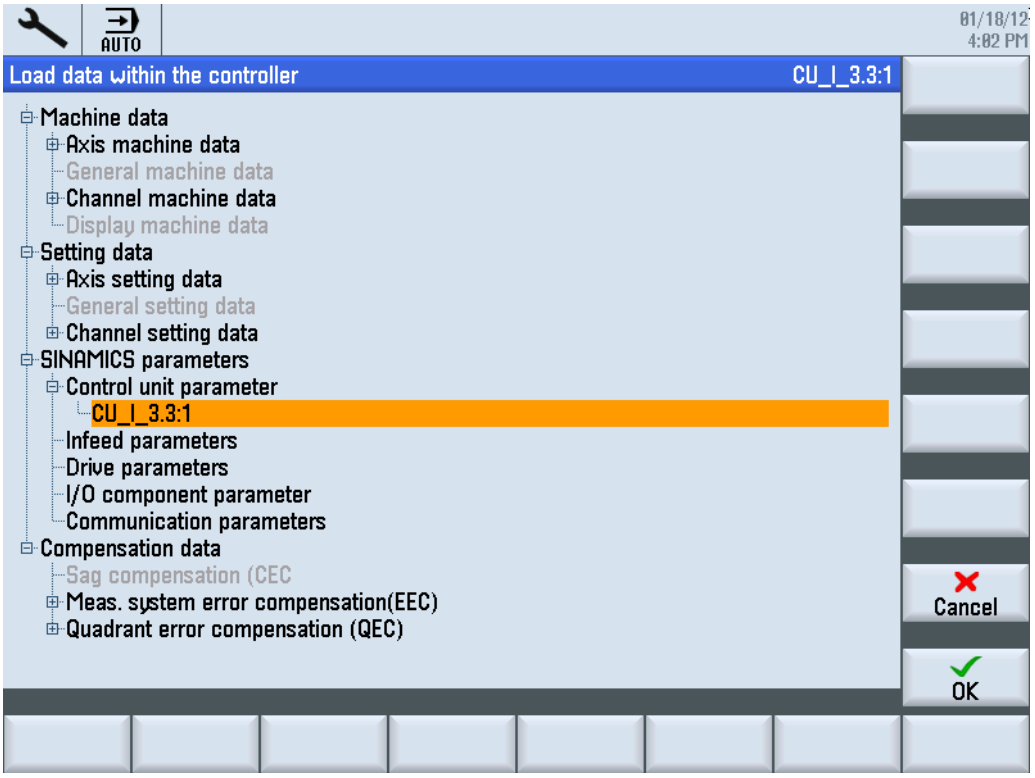



Figure 4-2 Managing data

## 4.7.1 How to transfer data within the controller

### Transfer data within the control

| NOTICE |
| --- |
| **Protection of the machine**<br>For safety reasons, the machine and setting data should only be transferred when the enable is locked. |

Procedure:

1. Select the "Transfer data within the control" option.
2. Select the source data in the data structure and confirm with "OK".
3. In the drop-down list, select an object, e.g. a different axis or a different drive object, to which you want to transfer the data and confirm with "OK".

4. Observe the safety instructions and check the enable signals at the machine and the drive.
5. For drive data, the "Load" softkey is used to transfer the data to the target object.

## 4.7.2 To save and load data

### Save data to a file

Procedure:

1. Select the "Save data to a file" option.
2. In the data structure, select the data that you want to save to a file and confirm with "OK".
3. Select as the storage location, a directory or a USB storage medium and enter a name.

**Note**

**SINAMICS parameters**

An ASCII file (*.TEA) is always generated when saving.

Three files of the following type are generated when saving the drive data:

- A binary file (*.ACX) that cannot be read.
- An ASCII file (*.TEA) that can be read or edited in the ASCII editor.
- A log file (*.log) that contains message texts (error situation) or is empty (successful storage).

### Loading data from a file

| NOTICE |
| --- |
| **Protection of the machine**<br>For safety reasons, the machine and setting data should only be transferred when the enable is locked. |

Procedure:

1. Select the "Load data from a file" option.
2. In the data structure, select the saved file and confirm with "OK".
3. In the drop-down list, select an object, e.g. a different axis or a different drive object, to which you want to transfer the data and confirm with "OK".
4. Observe the safety instructions and check the enable signals at the machine and the drive.
5. For drive data, the "Load" softkey is used to transfer the data to the target object.

## 4.7.3 How to compare data

### Compare data

You can select different data sources for the data comparison: Store the current data in files on the controller or data.

Procedure:

1. Select the "Compare data" option.
2. In the data structure, select the data that you want to compare.
3. Press the "Add to list" softkey to transfer the data to the list in the lower area of the display.
4. Press the "Delete from list" softkey to remove the data again.
5. If the list contains more than two data objects, you can activate the checkbox to compare two or more data objects from the list.
6. Press the "Compare" softkey to start the comparison. The display of the comparison results can take some time for extensive parameter lists.
7. Press the "Legend" softkey to display or hide the legend. The following display is the default:
   - Different parameters are displayed.
   - The same parameters are not displayed.
   - Parameters not available everywhere are displayed.

# 4.8 Friction compensation with adaptive characteristics

## Requirements

Function on the control in the commissioning operating area: Softkey "NC" → Softkey "Friction compensation". The "Friction compensation" dialog opens.

| Machine data | Meaning |
|---|---|
| MD32500 $MA_FRICT_COMP_ENABLE = 1 | Enable friction compensation for this axis |
| Friction compensation with adaptive characteristics: | |
| MD32490 $MA_FRICT_COMP_MODE = 3 | Injection time depends on the velocity setpoint |
| MD32490 $MA_FRICT_COMP_MODE = 4 | Injection time depends on the position controller output |



**Software option**

You require the following software option in order to use this function: Friction compensation

## Backup of the measurement results

If the friction compensation is activated for an axis and the option is not licensed, an alarm is output.

The measurement results are written to the machine data. They are changed as follows during the measurement series:

- When a measurement series is started with the "Start" softkey, the friction compensation values set in the machine data are reset.
- When the measurement series is terminated with the "OK" softkey, the new friction compensation values are in the machine data.
- When the measurement series is cancelled with the "Cancel" softkey, the previous friction compensation values are in the machine data again.

If the power fails during a measurement series, both the new and the previous friction compensation values are lost.

In order to avoid a data loss, it is recommended that you create a commissioning archive that contains the axis machine data of the last complete measurement series. You do not have to select the "With compensation data" option. For the SINUMERIK 828D, create a data class archive of the type "INDIVIDUAL" that contains the friction compensation values of the last complete measurement series.

## References

For more information, please refer to the Function Manual, Extended Functions, Chapter Compensations (K3).

## 4.8.1 Friction compensation with velocity injection pulse

### Operation on the SINUMERIK Operate

With this function, you can optionally traverse one or two axes. The measurement is graphically displayed as circle diagram. The values entered for the friction compensation only apply to the axis which is selected in the header line.

The scaling in the graphic display is adapted automatically as long as a scaling value is not entered manually. Up to nine measurements with different feedrates can be performed in one pass. With 1-axis and 2-axis traversing, the upper and lower reversal points are relevant for the tuning. The measurement consists of an initialization phase and the actual measuring process. The current status is displayed at the bottom right in the status line via a progress indicator.

If less than nine steps should be measured, overwrite any feedrates that are not required with zero. The following supplementary conditions apply:

- Feedrates with zero are permitted only at the end of the measuring series.
- All steps must be performed.

Procedure:

1. Press the "Axis +" or "Axis -" softkey to select the axis for which you activated friction compensation.

   The radius, the feedrate and the direction of rotation for the measurements are preset in the following setting data and apply for all axes in the channel. They can be adapted by pressing the "Data list" softkey:

| Designation | Unit | Machine data |
|---|---|---|
| Radius (linear axis)<br>Radius (rotary axis) | mm<br>° (degrees) | SD55820 $SCS_FRICT_OPT_RADIUS<br>SD55821 $SCS_FRICT_OPT_RADIUS_ROT |
| Feedrate (linear axis)<br>Feedrate (rotary axis) | [mm/min]<br>[°/min] | SD55822 $SCS_FRICT_OPT_FEED[0...8]<br>SD55823 $SCS_FRICT_OPT_FEED_ROT[0...8] |
| Direction of rotation (only active for two axes) | --- | SD55828 $SCS_FRICT_OPT_DIR |

2. To start the measurement, press the "Start" softkey. A part program is then automatically generated and selected by the NC; this is done by calling CYCLE790 which reverses the axis with the preset feedrate.

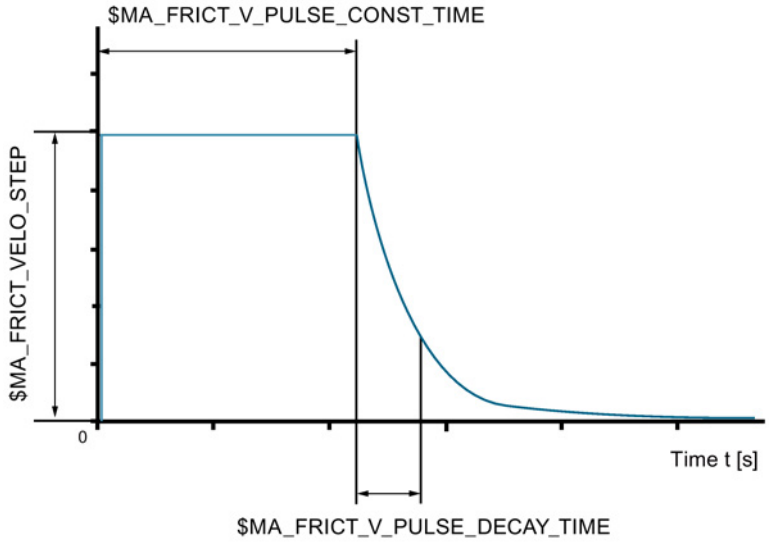
   Before the part program is started, a check is made as to whether the entered feedrates can be achieved based on the dynamic response of the axis. If this is not the case, a message is issued to reduce the feedrates.

3. The part program for the measurement series is then opened if you want to add further commands. Confirm with "OK" to close the window and continue.

4. The first step with the first feedrate of the measurement series is selected. Enter the friction compensation values in the following entry fields:

| Input field | Unit | Machine data |
|---|---|---|
| Amplitude | [mm/min]<br>or [°/min] | MD32571 $MA_FRICT_VELO_STEP |
| Decay time | [s] | MD32574 $MA_FRICT_V_PULSE_DECAY_TIME |
| Active time | [s] | MD32573 $MA_FRICT_V_PULSE_CONST_TIME |

To enter different compensation values for an axis depending on the traversing direction, select the "Set reversal points separately" option, otherwise the same value is taken for both traversing directions.




5. Press the "Next step" softkey to start a new measurement with the next feedrate and enter new values. If the measurement is repeated, the values appropriate for the feedrate from the last measurement are proposed.

After the last step, the measuring series is terminated with "OK" and the data is saved.

---

**Note**

**Incremental value change**

Use the following shortcut keys to change the compensation values incrementally:

- The selected entry field is activated with the <INSERT> key, and the color changes.
- The values are changed with different increments with the <Page Up> / <Page Down> keys and the ↑ ↓ cursor keys.
- End the input with the <INPUT> key.

---

**⇒ Further actions:**

- "Axis +" or "Axis -" softkey to select the axis for which the friction compensation is activated.
- "Start" softkey to start the measurement.

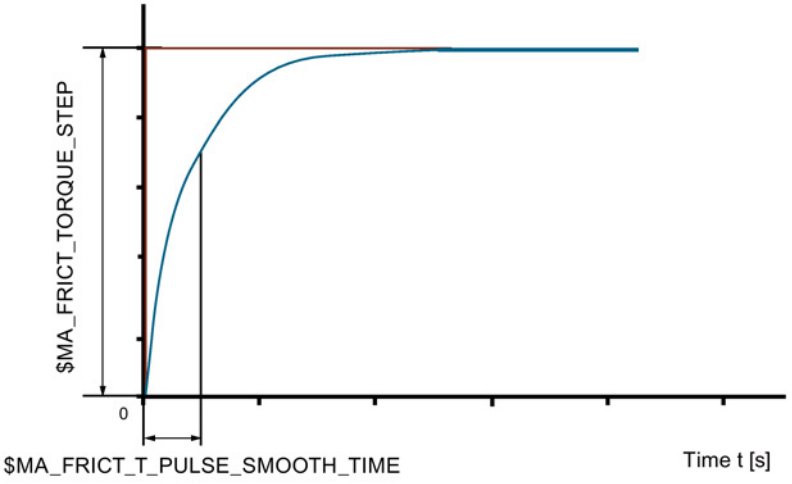
- "Next step" softkey to select the next feedrate.
- "Previous step" softkey to return to the previous feedrate.
- "Data list" softkey to switch to the list of machine data.
- "Cancel" softkey to cancel the recording and reject the entries.
- "OK" softkey to accept the values determined during the friction compensation.

## 4.8.2 Friction compensation with torque injection pulse

### Parameterizing machine data

The following machine data must be parameterized in order to use an acceleration-dependent torque injection pulse of the friction compensation:

| Designation | Unit | Machine data |
|---|---|---|
| Acceleration-dependent amplitude of the torque injection pulse | --- | MD32576 $MA_FRICT_TORQUE_STEP |
| Delay time of the torque injection pulse | [s] | MD32577 $MA_FRICT_T_PULSE_DELAY_TIME |
| Ramp-up time of the torque injection pulse | [s] | MD32578 $MA_FRICT_T_PULSE_SMOOTH_TIME |
| Weighting factor for the amplitude of the torque injection pulse | --- | MD32588 $MA_FRICT_T_STEP[0...9] |




Enter a factor between -1.0 and 1.0 in MD32588 $MA_FRICT_T_STEP[0...9] for each acceleration value in order to weight the amplitude of the torque injection pulse.

## 4.8.3 Data list

The machine data and setting data relevant for the friction compensation with adaptive characteristics is displayed in the data list:

- **Machine data:** All machine data can be edited.
- **Setting data:** Radius, feedrate and direction of rotation can only be changed **before** the start of the measurement series; no entries are possible during the measurement series.

⇒ **Further actions:**

Softkey "**<<**" to return to the "Friction compensation" dialog.

## 4.8.4 Example: Friction compensation for a linear axis

### Requirement

The machine data for activating the friction compensation is set for the X axis:

- MD32490 = 3
- MD32510 = 1

The X axis is measured with the following settings:

- SD55820 $SCS_FRICT_OPT_RADIUS = 2 mm radius
- SD55822[0...8] $SCS_FRICT_OPT_FEED  For the feedrate, see the following table

### Measuring the X axis

Procedure:

1. Select the dialog for the tuning via the "NC" and "Friction compensation" softkeys.
2. Select the X axis and press "Start".
3. Confirm the program editor by pressing "OK" without any further inputs. The X axis is not measured with separate reversal points.
4. Change the parameters until you are satisfied with the compensation.

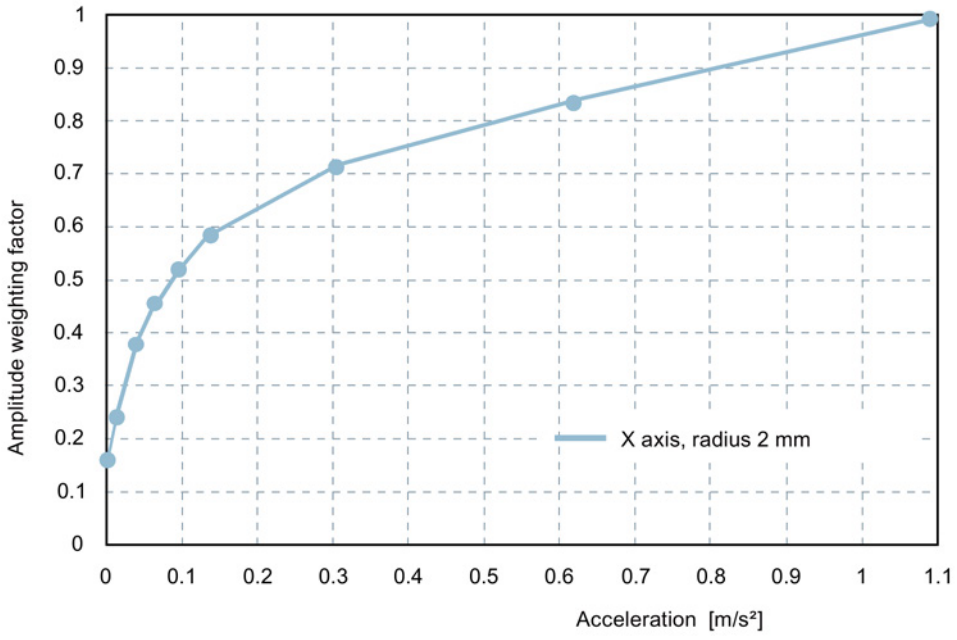
5. Press "Next >" to change to the next step.

| | |
|---|---|
| Step 1 of 9: 1,000 mm/min | Amplitude: 17.25 mm/min<br>Decay time: 0.043 s<br>Active time: 0.016 s |
| Step 2 of 9: 800 mm/min | Amplitude: 15.2 mm/min<br>Decay time: 0.046 s<br>Active time: 0.014 s |
| Step 3 of 9: 600 mm/min | Amplitude: 14.050 mm/min<br>Decay time: 0.026 s<br>Active time: 0.016 s |
| Step 4 of 9: 400 mm/min | Amplitude: 11.250 mm/min<br>Decay time: 0.030 s<br>Active time: 0.016 s |
| Step 5 of 9: 200 mm/min | Amplitude: 7.500 mm/min<br>Decay time: 0.035 s<br>Active time: 0.016 s |
| Step 6 of 9: 100 mm/min | Amplitude: 4.900 mm/min<br>Decay time: 0.050 s<br>Active time: 0.016 s |
| Step 7 of 9: 1,500 mm/min | Amplitude: 22.175 mm/min<br>Decay time: 0.027 s<br>Active time: 0.018 s |
| Step 8 of 9: 2,100 mm/min | Amplitude: 25.000 mm/min<br>Decay time: 0.015 s<br>Active time: 0.018 s |
| Step 9 of 9: 2,800 mm/min | Amplitude: 30.000 mm/min<br>Decay time: 0.010 s<br>Active time: 0.016 s |

6. Repeat this action until you have finished step 9.
7. Press "OK" to confirm and save the data.

## Result

The following graphic shows the characteristic curve with the adapted amplitude value for the associated acceleration in a monotonically increasing series:



# Commissioning the drive

# 5

## 5.1 Commissioning the drive

### 5.1.1 Sample configuration – overview

#### Configuration overview

The principle sequence of the commissioning with SINUMERIK Operate is the same for SINAMICS S120 Combi and SINAMICS S120 Booksize.

The procedure when configuring is therefore illustrated using the following examples:

**Example 1:** SINAMICS S120 Combi without NX:

- The infeed unit is automatically configured.
- Spindle with 1PH8 motor is configured.
- DO4 with two encoders: here, encoder 2 must still be configured: "User defined"
- DO6 with two encoders, both are automatically detected.

**Example 2:** Configuring the infeed for SINAMICS S120 Booksize

#### Configuring the drive

The DRIVE-CLiQ connections must correspond to the topology rules in Section "Topology rules for DRIVE-CLiQ (Page 144)". The sequence of the DRIVE-CLiQ connections corresponds to the sequence of the SINAMICS drive object numbers (= default setting):

| Axes | SINAMICS drive object | |
|---|---|---|
| | Number (DO) | Name |
| --- | 1 | Control Unit |
| --- | 2 | Line Module |
| MSP1 | 3 | SPINDLE |
| MX1 | 4 | SERVO 1 |
| MY1 | 5 | SERVO 2 |
| MZ1 | 6 | SERVO 3 |
| MA1 | 7 | SERVO 4 |
| --- | 11 | HUB module |
| --- | 9 | TM54F master |

## Sample configuration

The following examples refer to the following configuration:

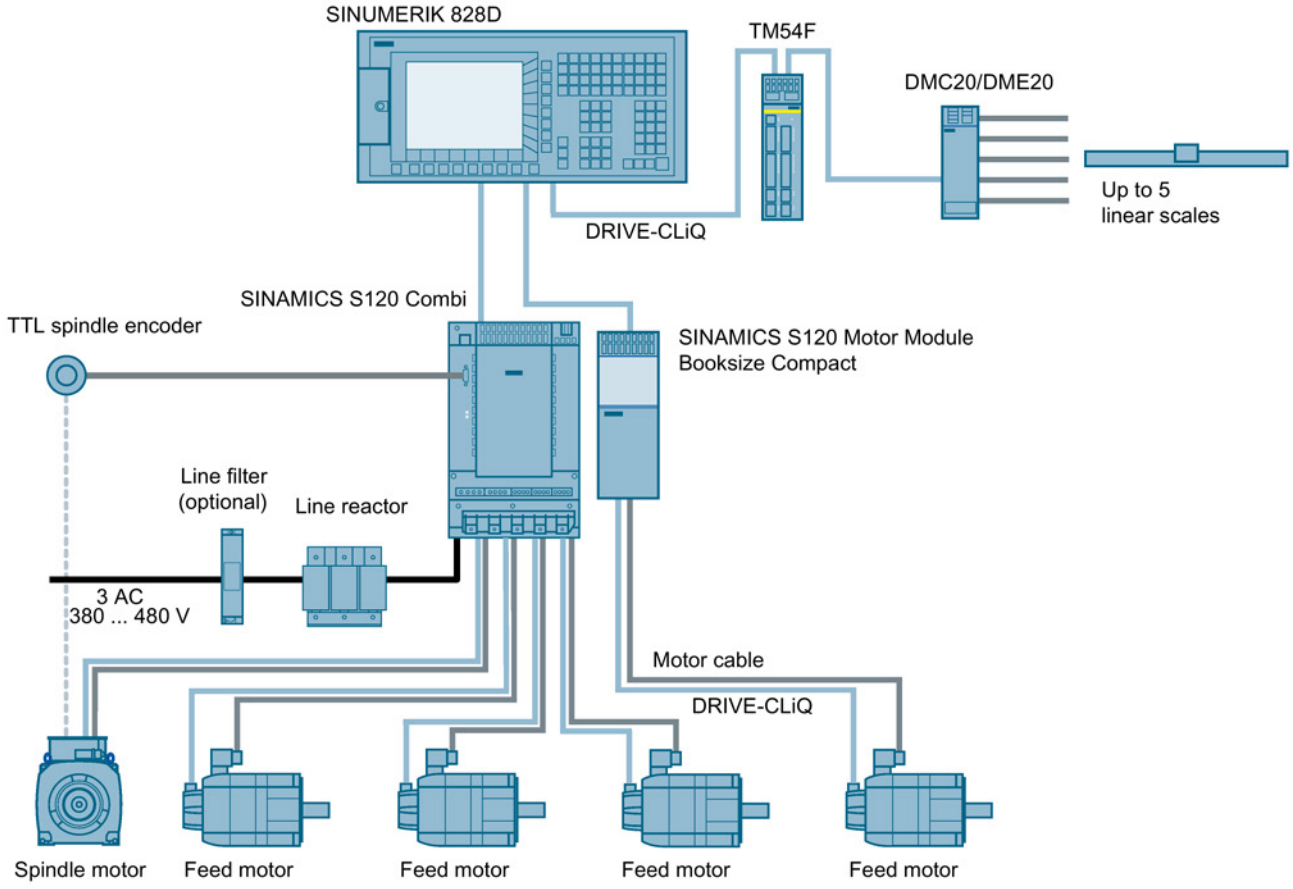


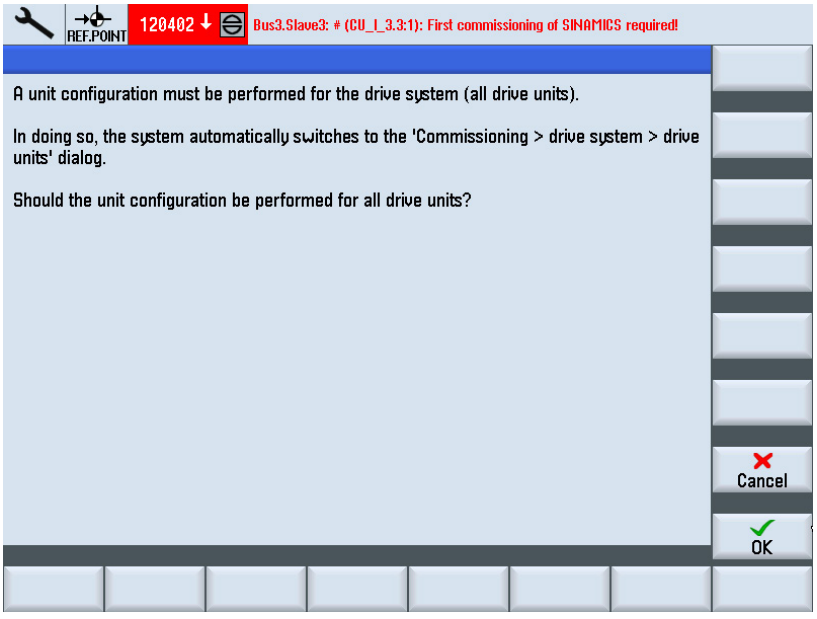

Figure 5-1 Example of the commissioning

## 5.1.2 Automatic first commissioning

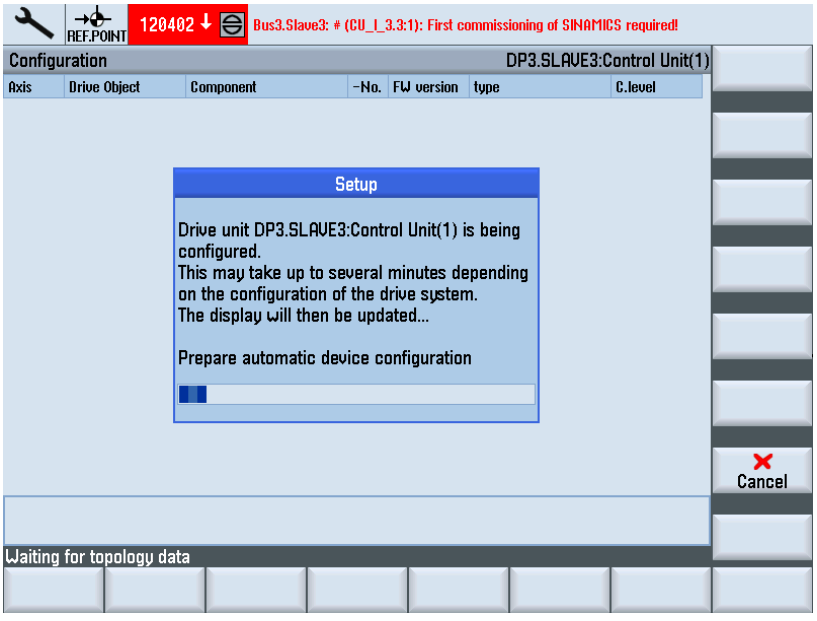
### Automatic first commissioning

Procedure:

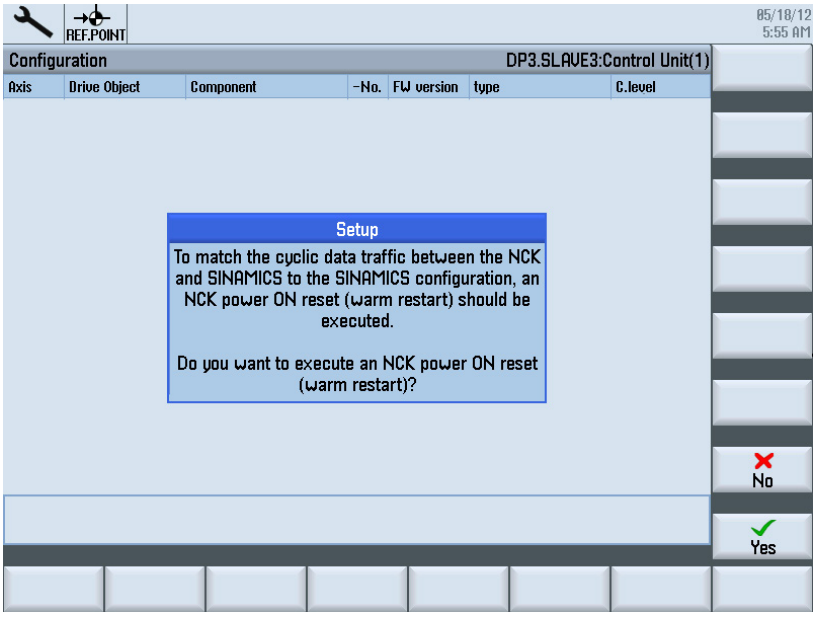
1. After the complete drive system has powered-up, a dialog box for automatic device configuration is displayed: "A device configuration must be performed for the drive system (all drive devices."



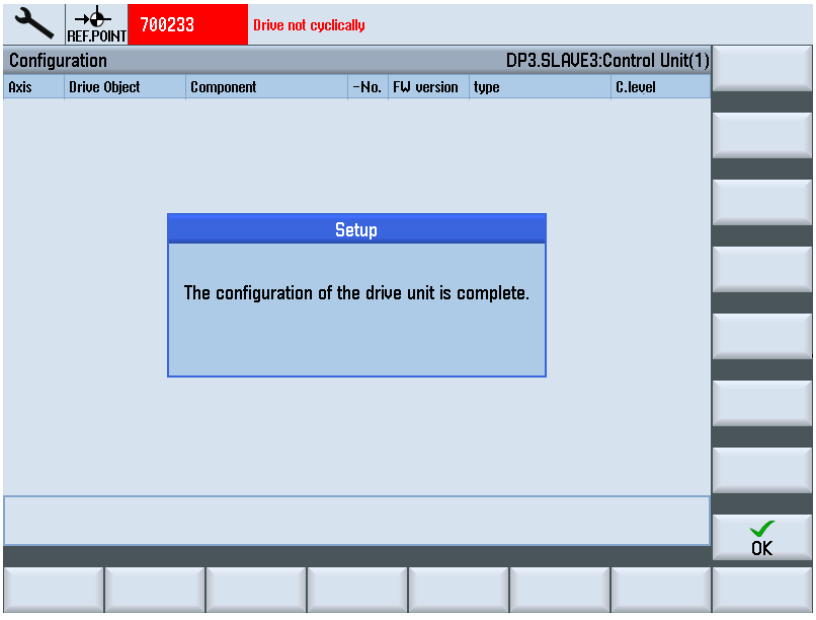

2. Select "OK" in order to start the automatic device configuration. The system topology is then automatically read out. This operation can take several minutes.



3. The system then requests a warm start. Confirm with "Yes". This operation can take several minutes.




4. After the warm start you will receive an overview of the topology. Confirm with "OK" to continue the commissioning.



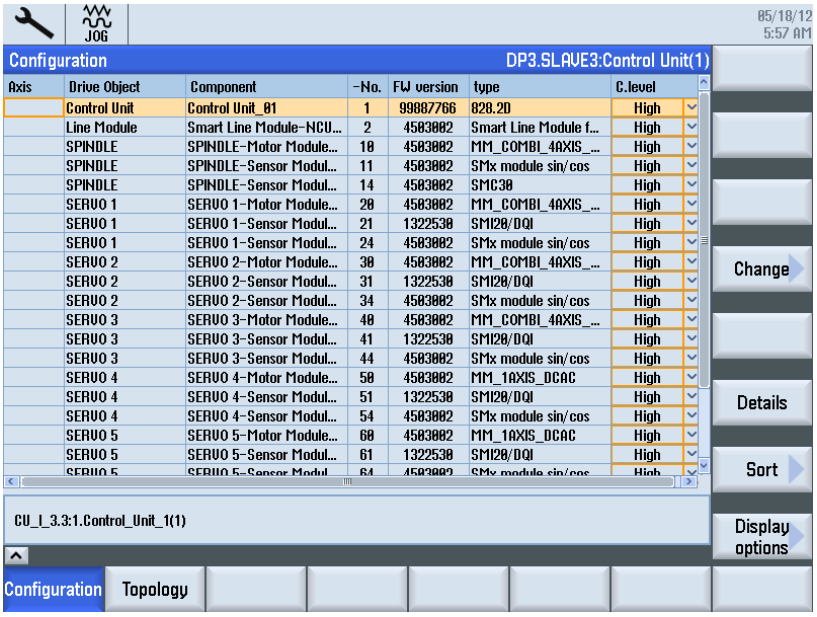

**Note**

**Establishing a defined initial state**

If the automatic device configuration does not start then it has already been implemented: To establish a defined state or to start once more, please load "Factory settings". After that, a defined output state is present again.

## Configuration after the first commissioning

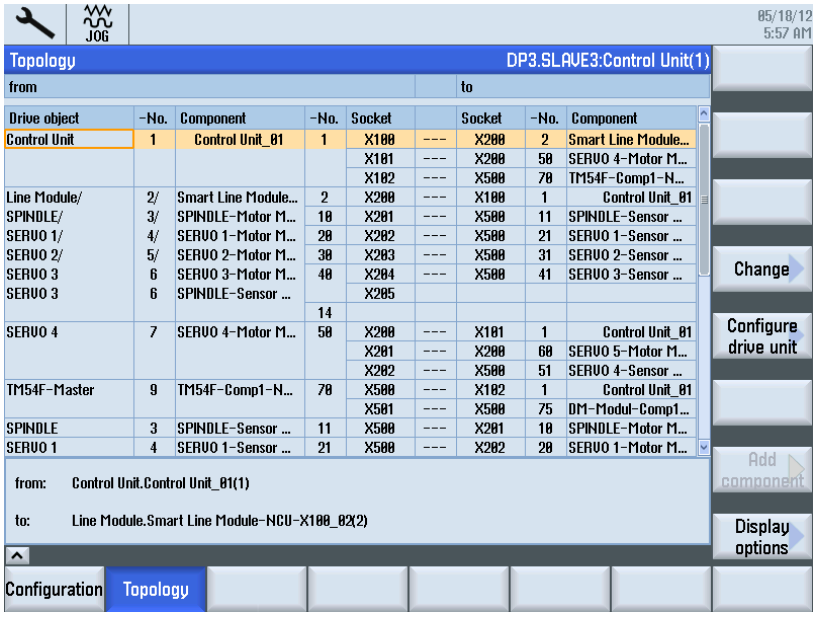
In the "Configuration" window you can obtain a list of the connected components for the selected drive unit. You can identify the assignment of the components to the drive objects, e.g. the assignment of the Motor Module, motors and encoders to the drive objects and their assignment to the NC machine axes (after the axis assignment).



## Checking the topology after the first commissioning

In the "Topology" window you can see the DRIVE-CLiQ wiring of the components for the selected drive unit. The connected component connections are arranged directly next to one another and shown in a table. This means that you can follow the wiring of every DRIVE-CLiQ line starting from the Control Unit or NX expansion module (e.g. up to an encoder).

The wiring of the drive objects "from/to" is displayed below the table.



## 5.1.3 Example: This is how you check the infeed configuration (S120 Combi)

### Checking the infeed configuration

The drive object was commissioned. The automatically determined characteristic data are displayed in the following overview (two pages): Data for the line filter, Voltage Sensing Module (VSM) and for the line voltage.

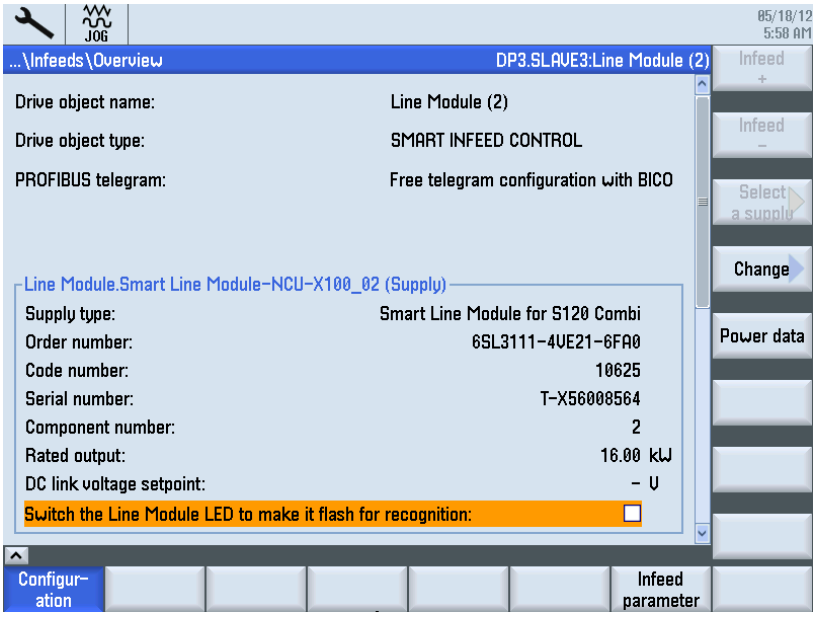



Figure 5-2 S120 Combi infeed (page 1)

The device connection voltage entered in the "Network data" dialog is displayed: This is the basis for monitoring the supply voltage (p0281 - p0283) for the alarm threshold and shutdown threshold.

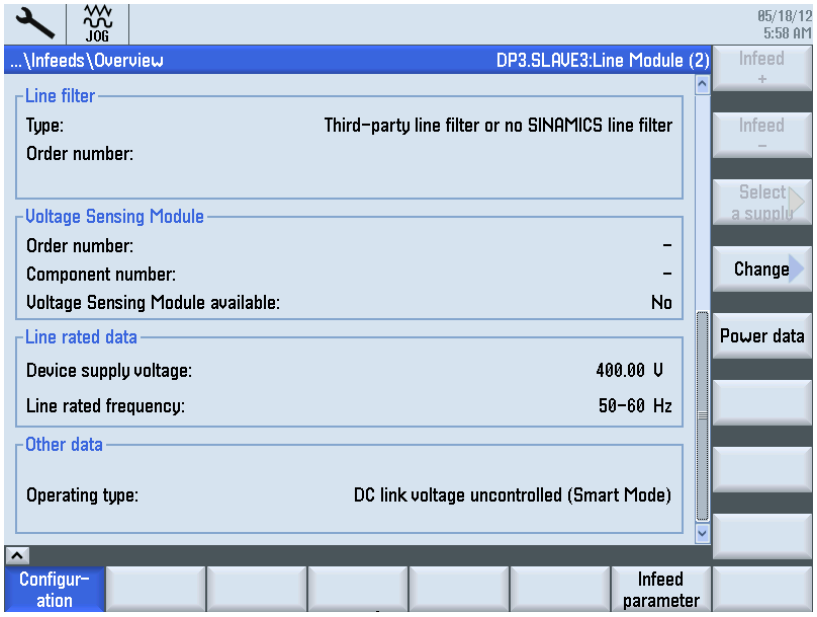



Figure 5-3 S120 Combi infeed (page 2)

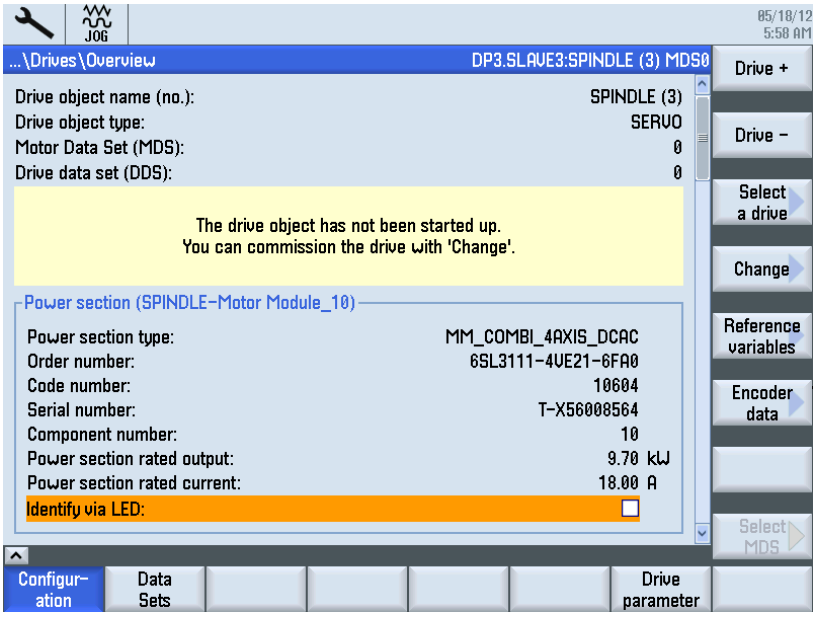
## 5.1.4 Example: This is how you configure the spindle (DO3)

### Configuring the spindle

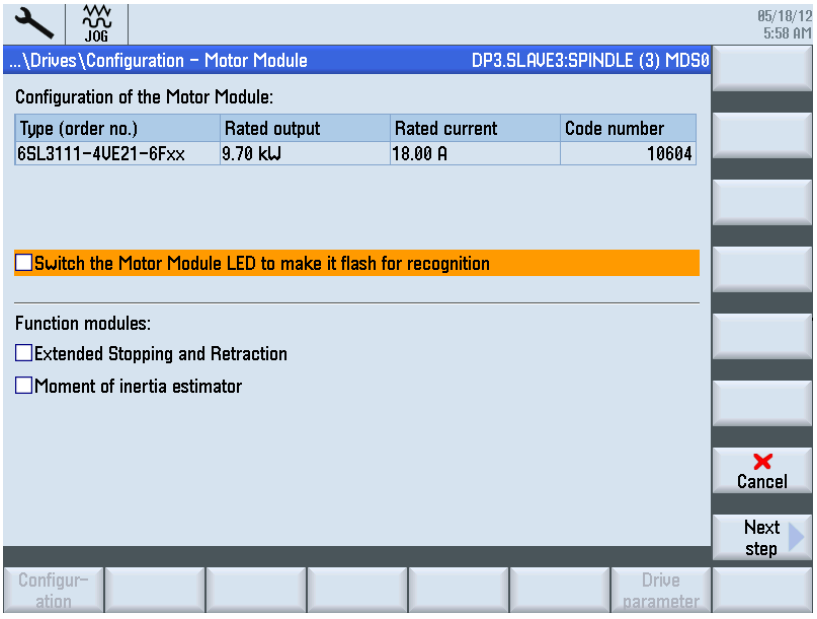
The spindle (drive object DO3) must be manually commissioned, because it is not operated with an SMI motor. As a consequence, the motor and measuring systems cannot be automatically determined.

Procedure:

1. Select "Change" to configure the spindle:



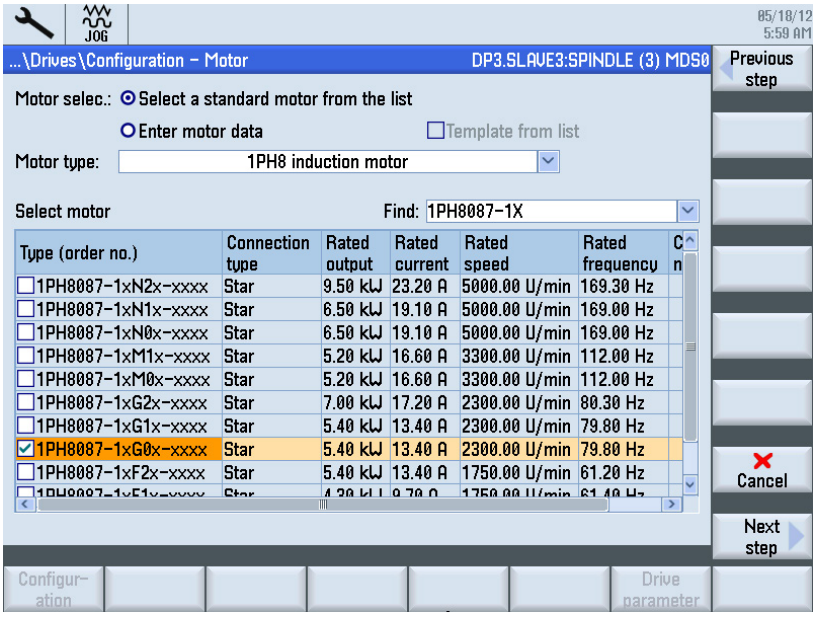

2. The characteristic data of the automatic device configuration of the drive object (DO3) is displayed:




Optionally, you can activate the function modules, which in some cases require a license. This way, further parameters become active, which are required for commissioning the relevant function module.

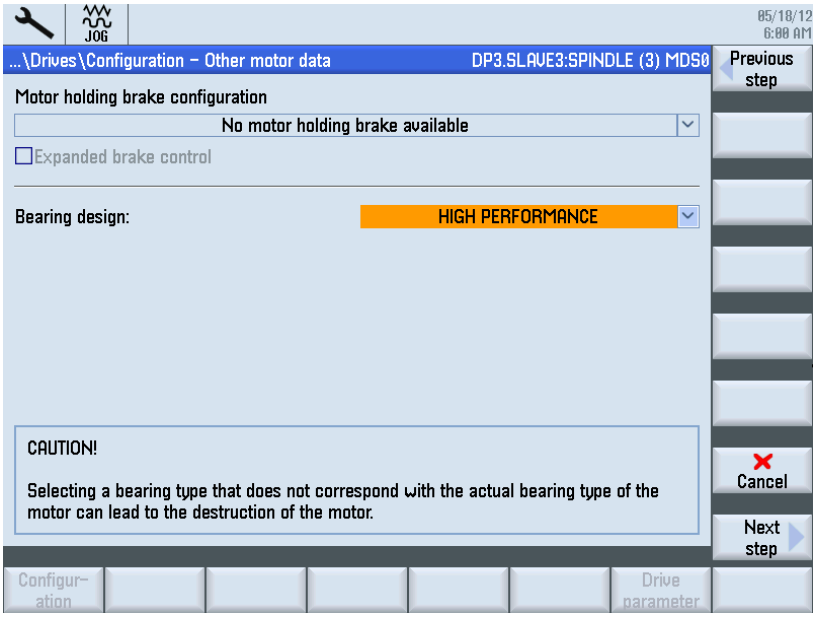
3. Select "Next step" >. To configure the motor, select "Select standard Siemens motor from the list":

   First select the motor type. In order to restrict the list for selecting the motors, enter text in the "Search" field which works as a filter. The character * is used as a wildcard. The text works for all column contents in the list except for units and connection type.



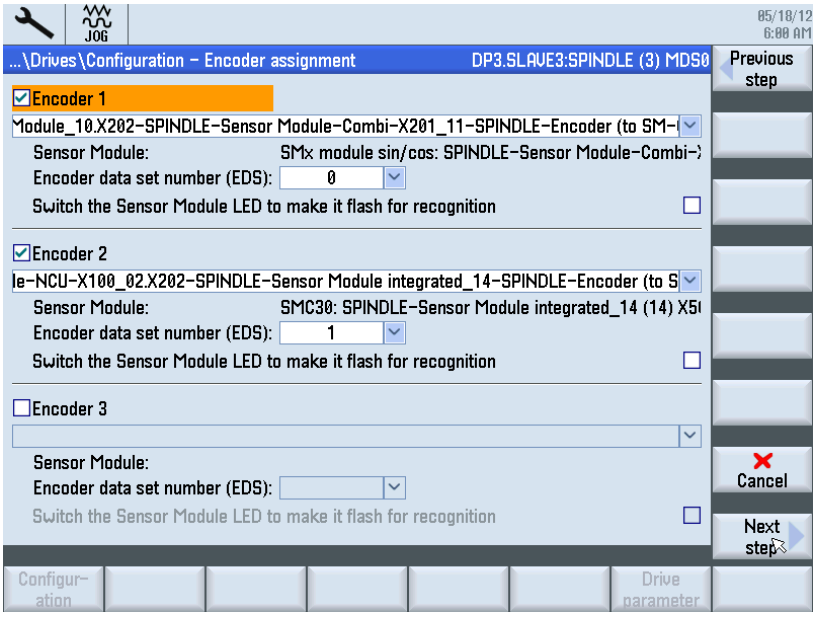

4. Select "Next step" >. "Bearing version" is only listed if this motor is available with different bearing versions.

   You can select the bearing type here: This limits the speed to the maximum speed of the bearing in order to prevent mechanical destruction. When selecting "No specification" the maximum speed of the bearing is not monitored.

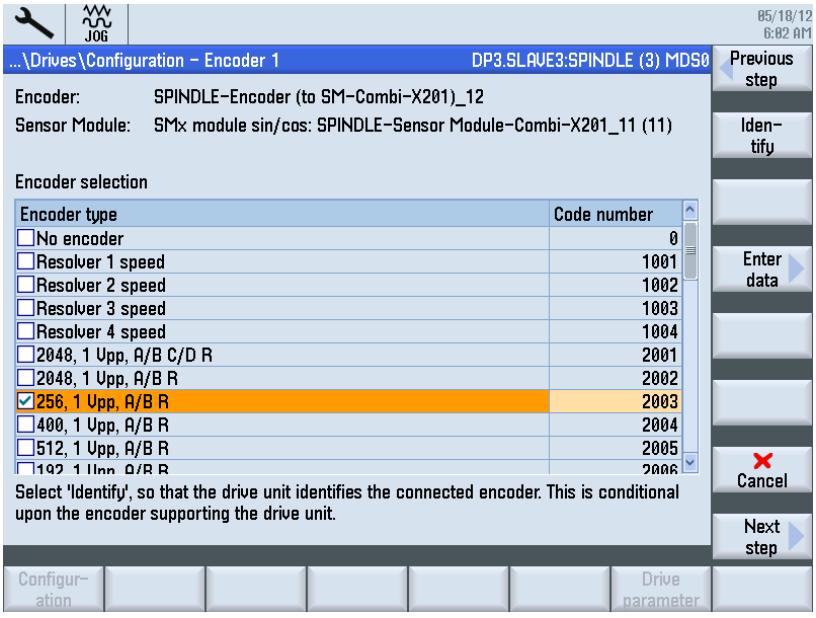


5. Select "Next step" >. Here you can assign a maximum of three encoders to the selected SERVO:

   If the Sensor Module (SMx) is already inserted correctly on the SERVO, then it is automatically assigned here. More than one measuring system can also be assigned automatically.



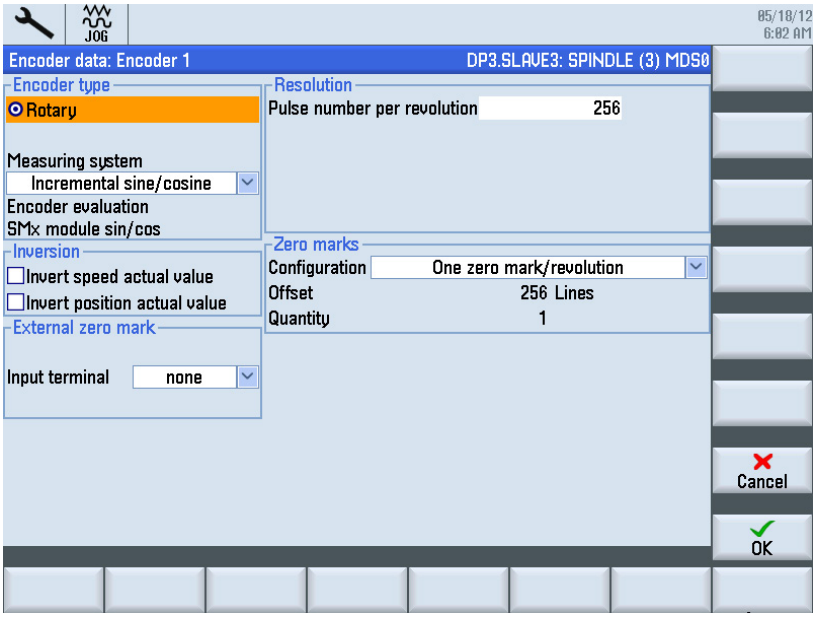
6. Select "Next step" >.



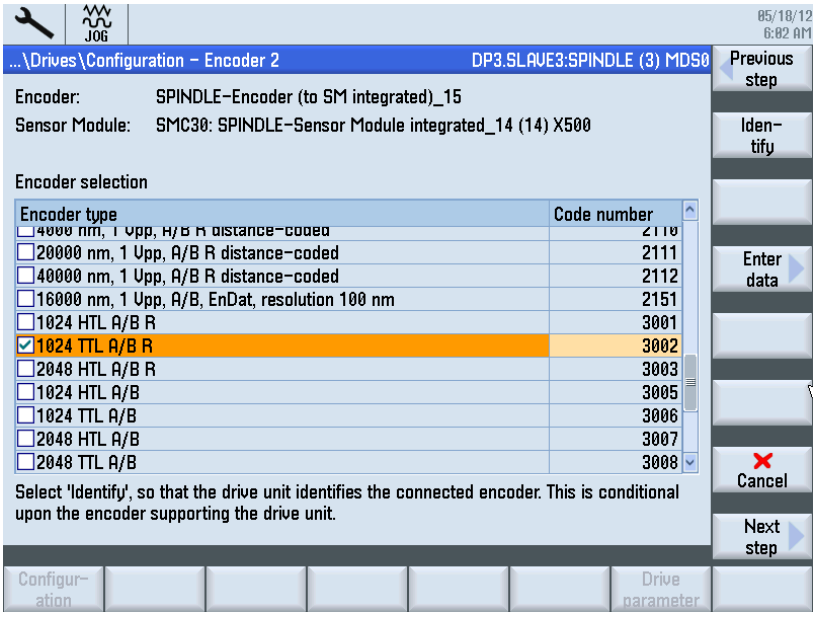

The name of the encoder and the Sensor Module are displayed here via which the encoder is connected.

- **Case 1:** Automatically selected with EnDat and DRIVE-CLiQ, no further settings required. To check the default settings, press the "Enter data" softkey.
- **Case 2:** Encoder was not automatically detected. If the connected encoder is in the list, select the encoder based on the name/code number. Otherwise, select a similar encoder from the list to enter the encoder data.

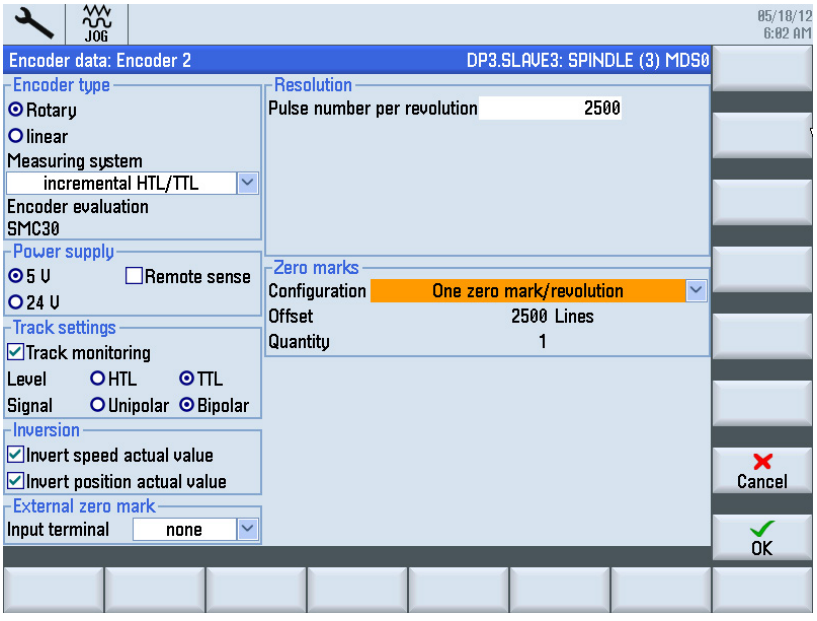
7. Select "Enter data >".



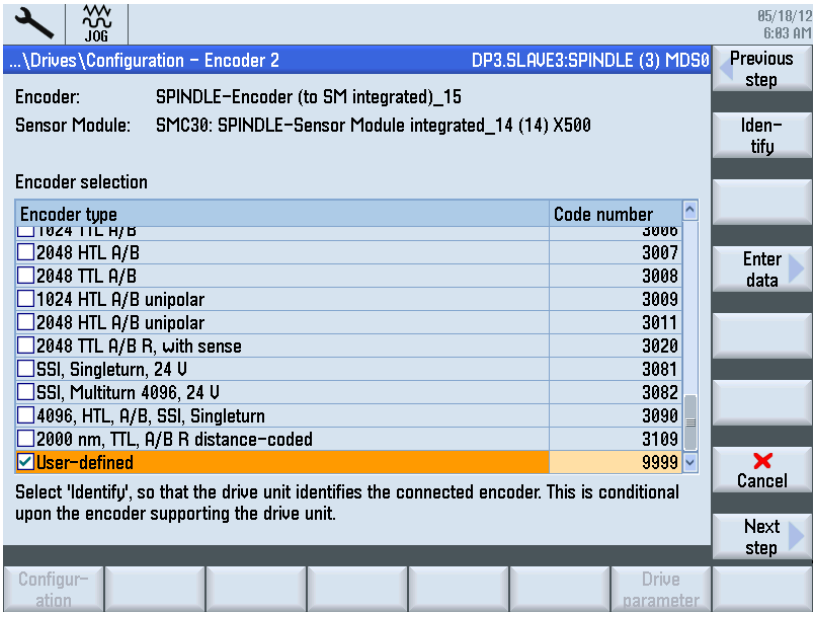
8. Confirm the encoder data with "OK".




9. Select "Enter data >".

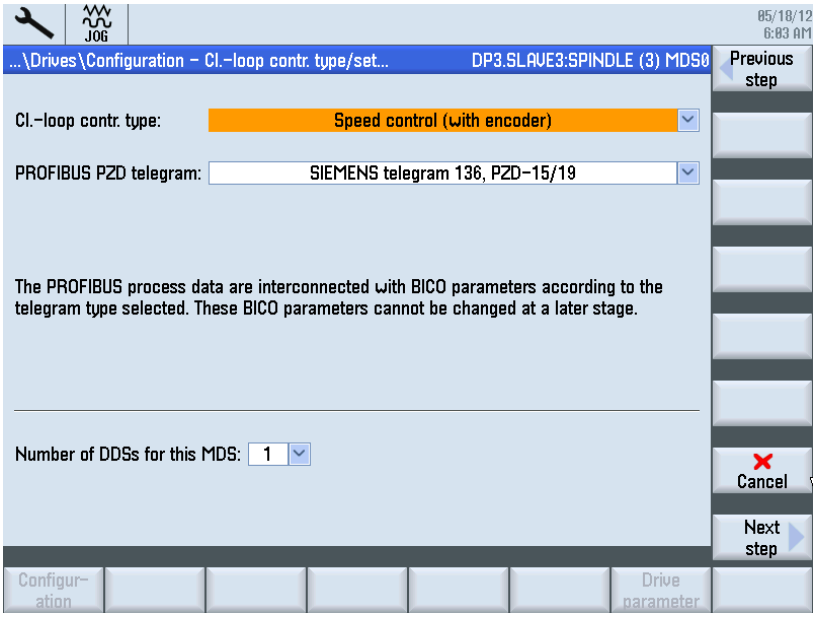


10. Confirm the encoder data with "OK".




11. Select "Next step" >.

Default setting: The control type, PROFIBUS telegram and the number of DDS are generally pre-assigned correctly.



Here you set the control type and the PROFIBUS telegram type as well as the number of DDS.

– Control mode:

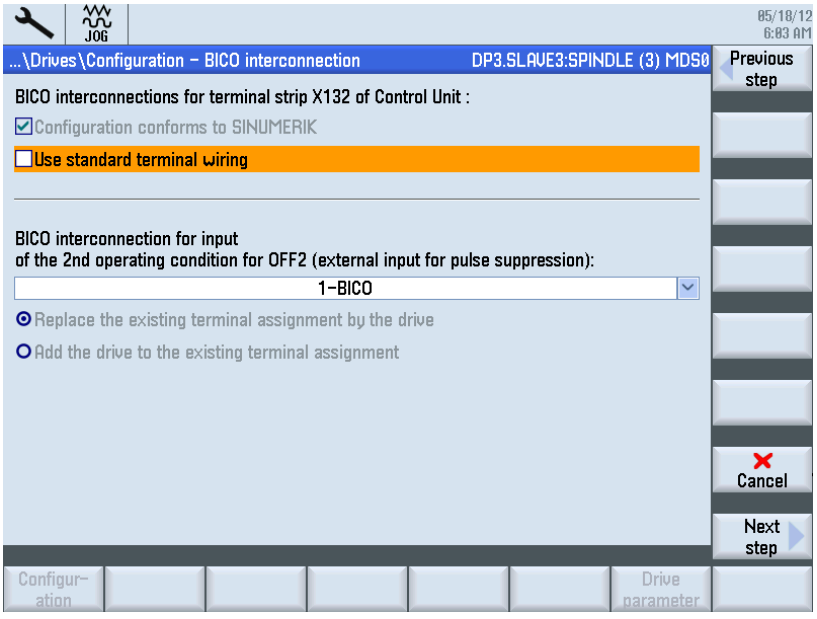
  If no motor encoder is assigned to the SERVO, then "speed control (without encoder)" is automatically set as the control type.

  If a motor encoder is assigned to the SERVO, then "speed control (with encoder)" is automatically set as the control type.

– Telegram type: The default setting is telegram 136: In the event of a change in the telegram type, the PLC configuration must also be adapted.

– Data set: The default number for the DDS is 1. If a drive data set switchover is required, then more DDS can be set up if necessary. Points to bear in mind: A higher number of DDS means that it takes the SINAMICS system longer to boot. Further DDS and MDS can also be set up subsequently in the data set Wizard.

12. Select "Next step" >.




Make the following selection with both of these checkboxes:

– **Default setting: "**SINUMERIK-compliant interconnection for the terminal wiring"

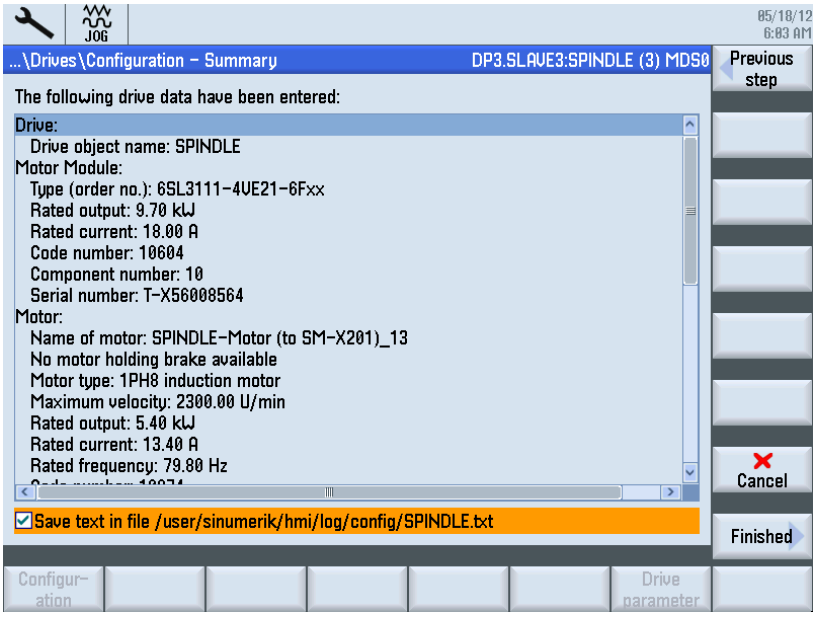
  This means that the BICO interconnections for the terminal wiring are set in such a way that all enables and responses communicate correctly.

  If the option is not activated (check box is not selected), then changes have been made in the BICO interconnection in relation to the standard interconnection. If these changes were not intended you can use the "Set standard terminal wiring set" option to restore the SINUMERIK compliant interconnection.

– **Option: "**Set BICO connections for the standard terminal wiring"

  Enable this option in order to set the standard terminal wiring. This means that the configuration is interconnected in compliance with SINUMERIK.

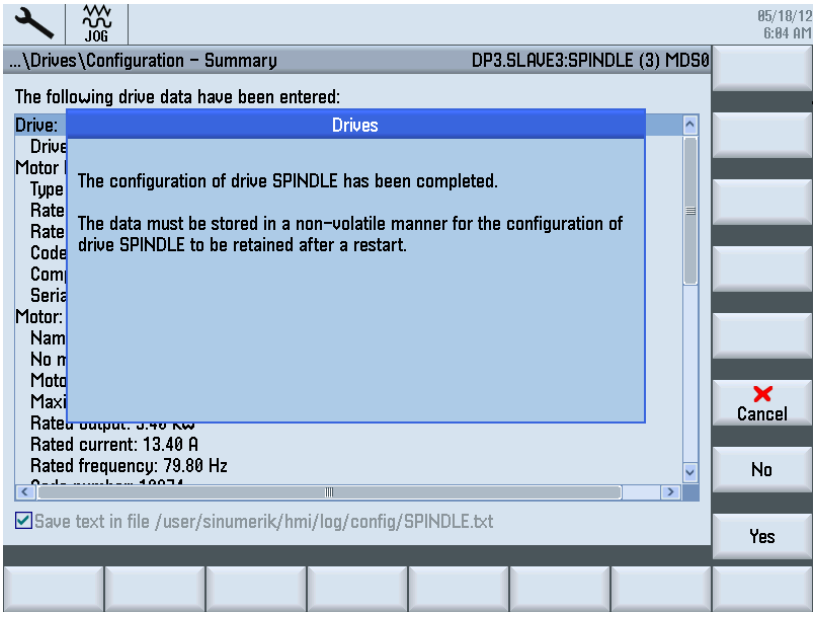
13. Select "Next step" >.




All data with which the drive object has been configured is displayed in the summary.

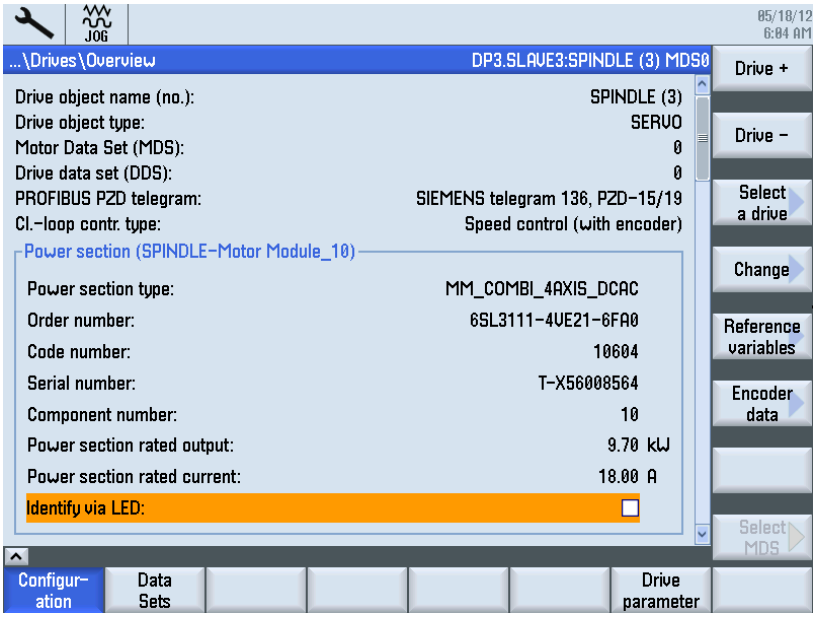
Option: Activate "Save text to file" to save the configuration data in a text file.

14. Select the "Finish >" softkey to exit the commissioning of this drive object. In case of changes, a query to save is output.




15. To ensure that the data after power ON/OFF is retained, confirm with "Yes".

    The data is saved to the CompactFlash card and installed in a commissioning archive with drive data.




The spindle has now been commissioned.

## 5.1.5 Example: This is how you configure an encoder for an axis (DO4)

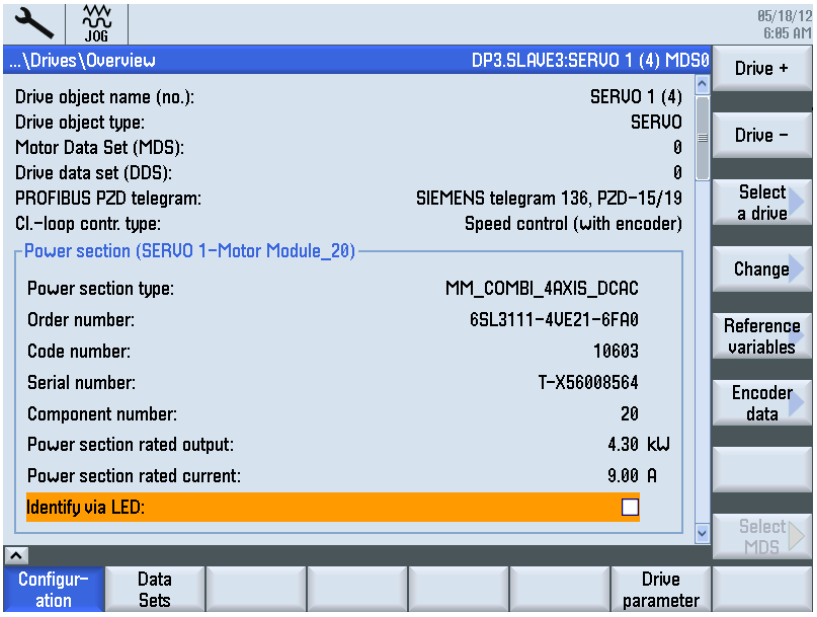
### Configuring encoder 2 for the axis

Drive object DO4 was commissioned with the automatic device configuration.

- Encoder 1 was detected.
- Encoder 2 is not in the selection list and must still be configured.

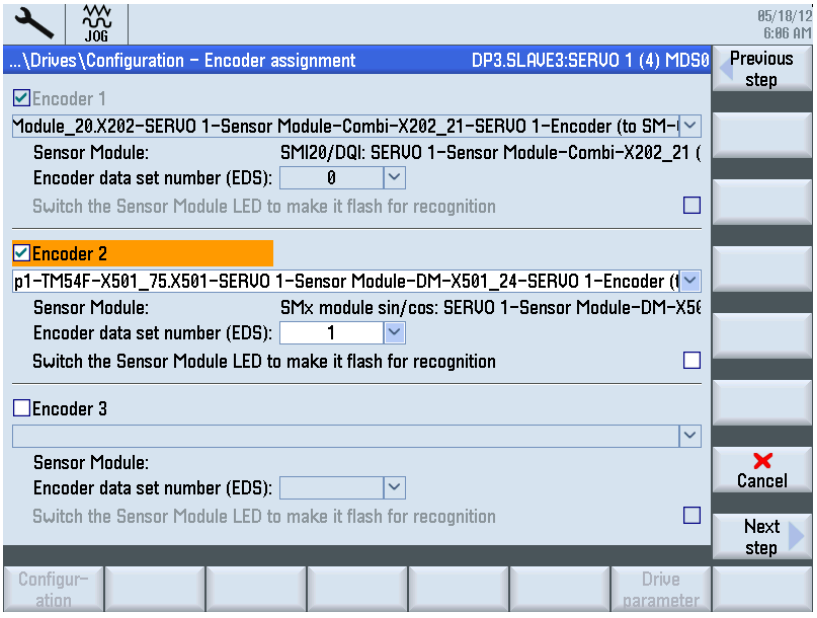
Procedure:

1. This is demonstrated using the following example:



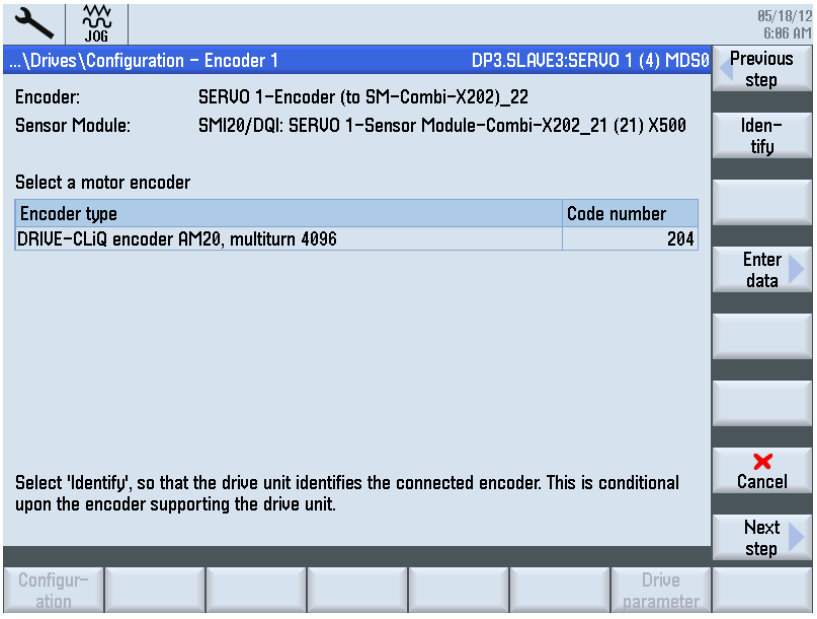

2. Select "Change >".

3. Select "Next step >" until you have accessed the "Encoder assignment" dialog:




4. Select "Next step >: Encoder 1 was detected.

   The name of the encoder and the Sensor Module are displayed here via which the encoder is connected.




**Note**

**Encoder identification**

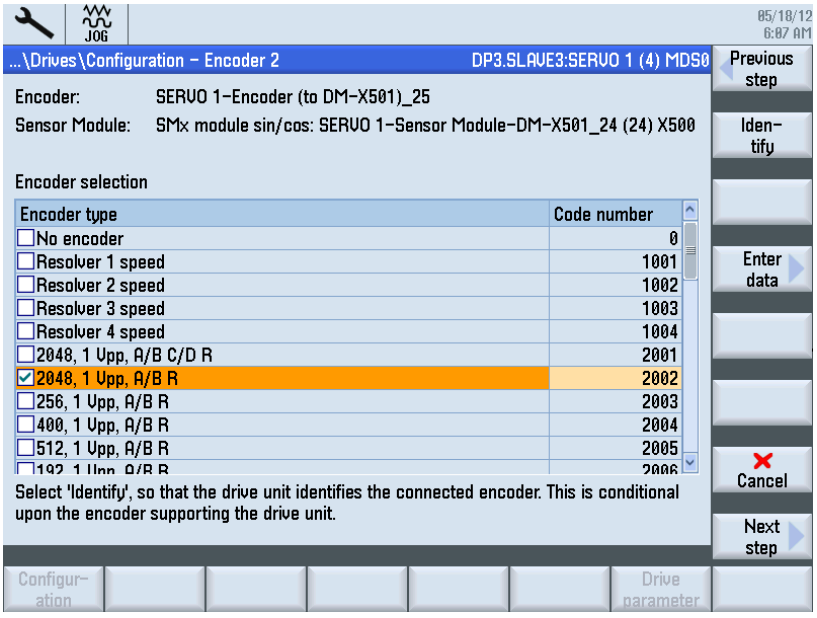
With the "Identify" softkey restart the identification of the encoder in order to check whether the encoder displayed matches the encoder that is actually connected. The encoder can only be identified if it is an absolute encoder or a DRIVE-CLiQ encoder.

5. Select "Next step" >.

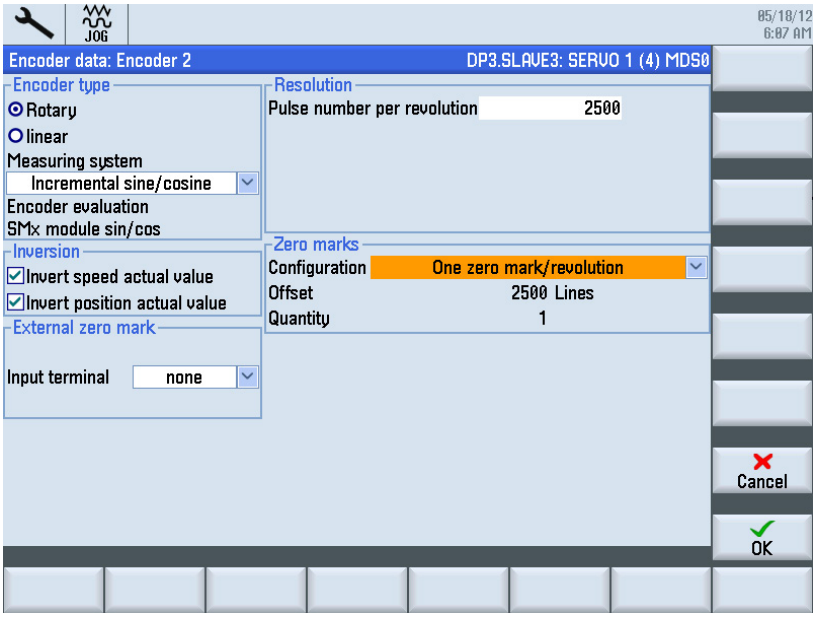
   Encoder 2 was not automatically detected ("no encoder") and must still be configured:

   If the connected encoder is in the list, select the encoder based on the name/code number. Otherwise, select a similar encoder from the list and press the "Enter data" softkey to enter the encoder data.

6. Select the following encoder type: Code number 2002.




7. Select "Next step >" and enter the encoder data:

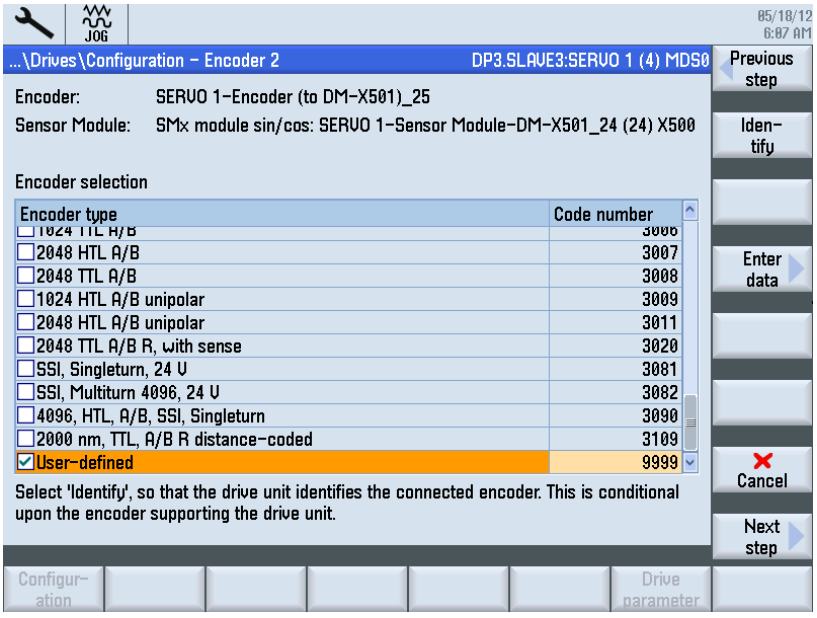



Inverting: Inverting the speed and position actual value.

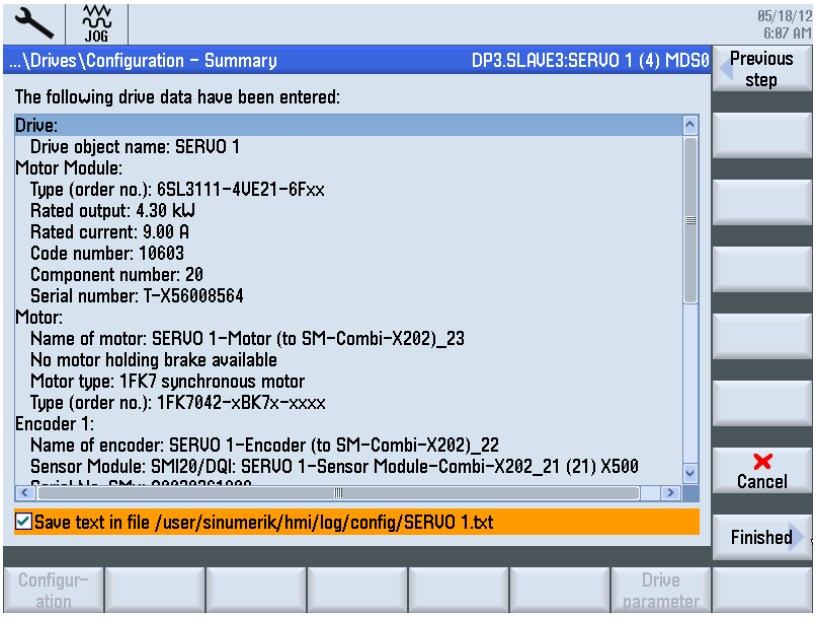
Pulses per revolution: 2500

Confirm with "OK".

8. Encoder 2 is configured and is displayed as "User-defined".

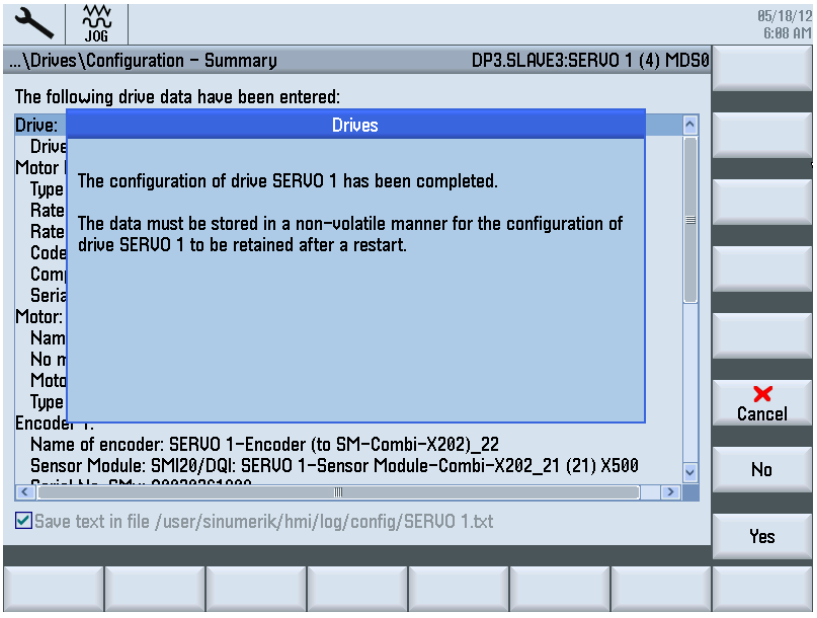



9. Select "Next step >" until the following summary is displayed:




Option: Activate "Save text to file" to save the configuration data in a text file.

10. Select "Ready >" to exit the configuration.




11. Select "Yes" to save the configuration in a non-volatile fashion.

The axis has now been commissioned.

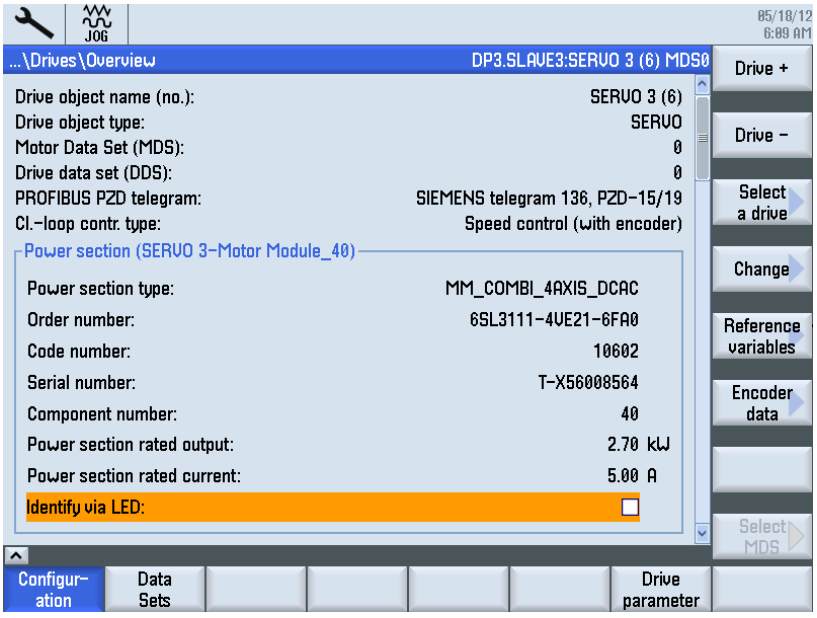
## 5.1.6 Example: This is how you check the configuration of an axis (DO6)

### SERVO DO6: Checking the encoder

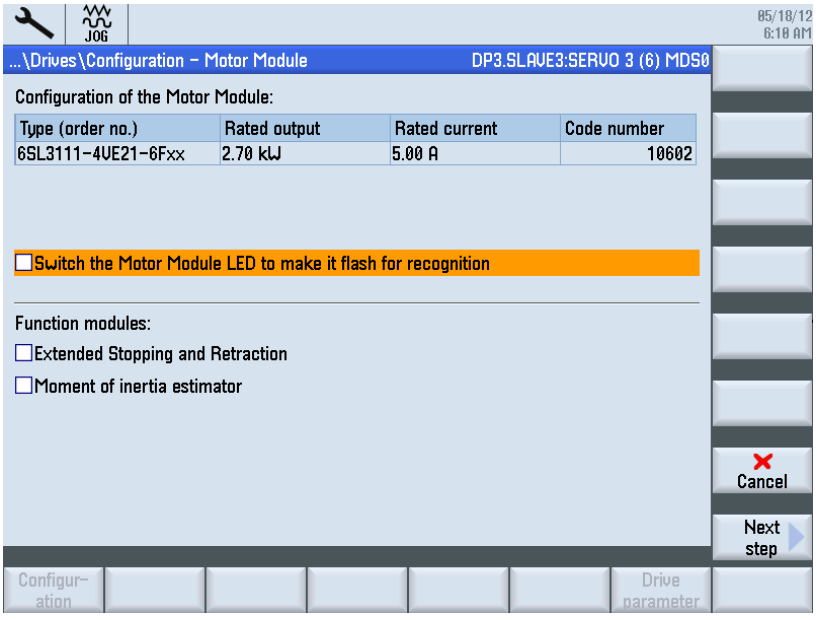
Drive object DO6 was commissioned with the automatic device configuration. Both encoders are in the selection list and can be identified.

Procedure:

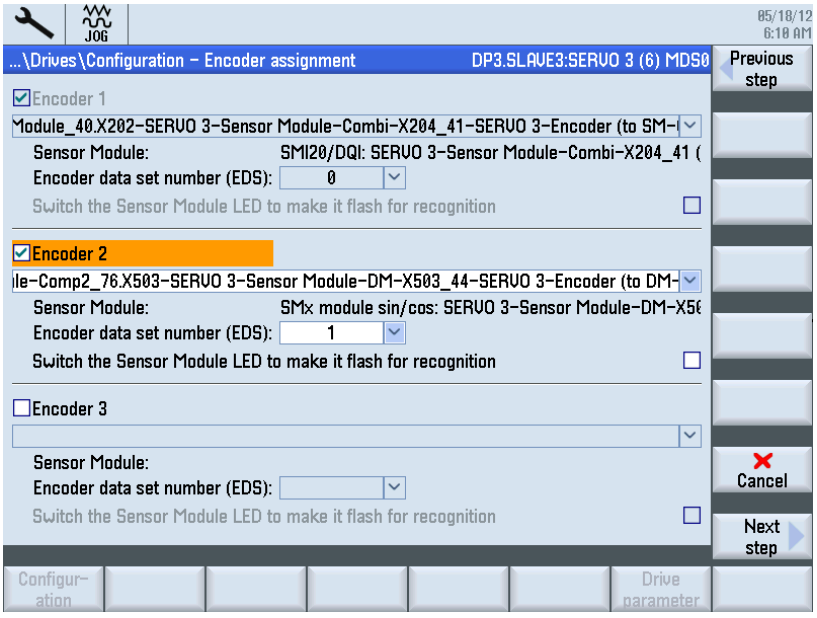
1. This is demonstrated using the following example:




1. Select "Change >".

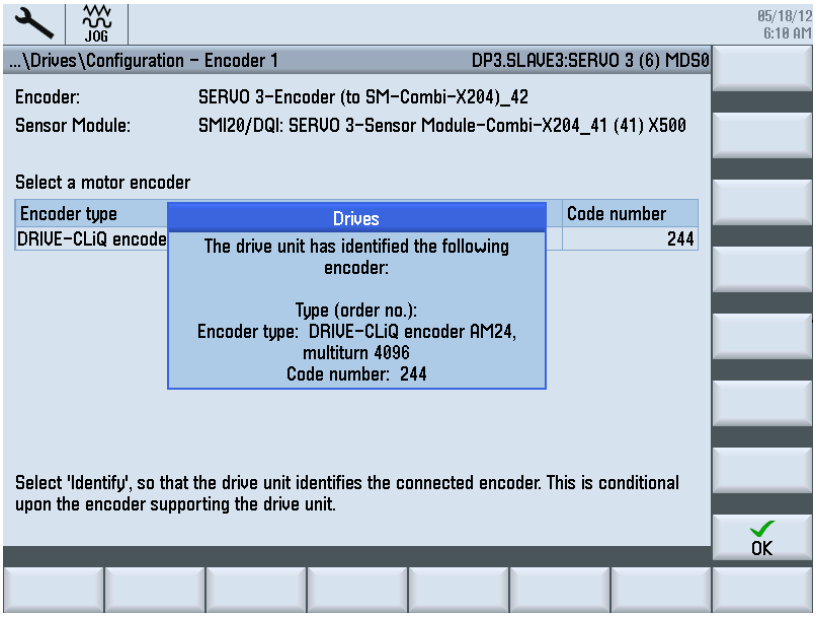


2. Select "Next step >".



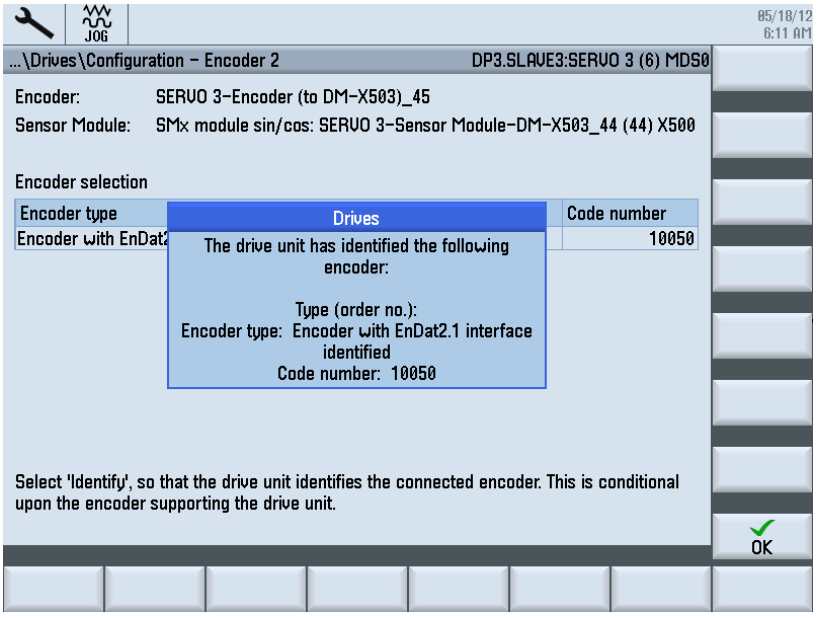

In this case, the Sensor Module (SMx) is already inserted correctly on the SERVO and it is automatically assigned. More than one measuring system can also be assigned automatically.

3. Select "Next step >".
4. Press the "Identify" softkey for encoder 1.

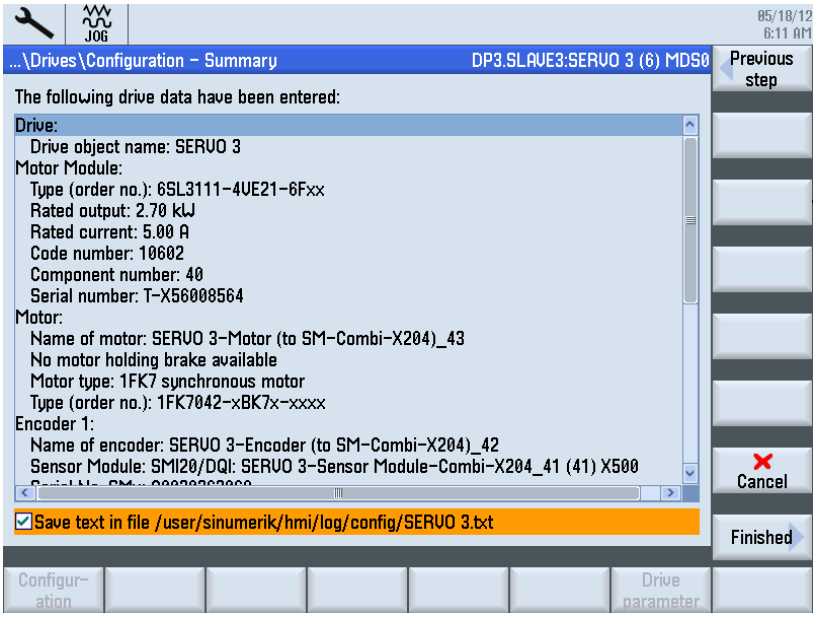



5. Select "Next step >".

6. Press the "Identify" softkey for encoder 2.

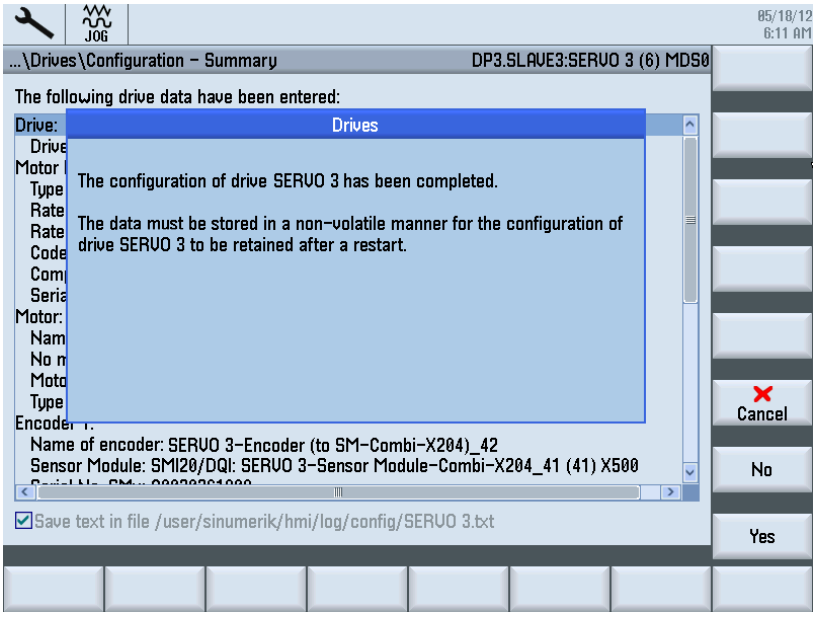



7. Confirm with "OK" and select "Next step >" until the following summary is displayed:




Option: Activate "Save text to file" to save the configuration data in a text file.

8. Select "Ready" to exit the configuration.




The axis has now been commissioned.

## 5.1.7 Example: This is how you configure the infeed (S120 Booksize)

### Configuring the infeed (S120 Booksize)

Procedure:

1. Select "Change" to configure the infeed:

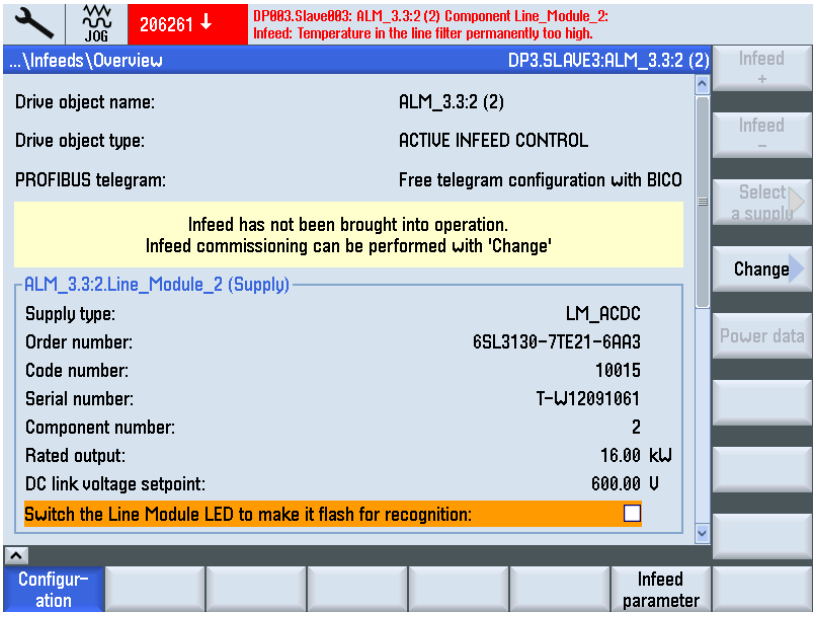



Figure 5-4 Infeeds > Overview

2. The characteristic data of the infeed detected with the automatic device configuration is displayed:

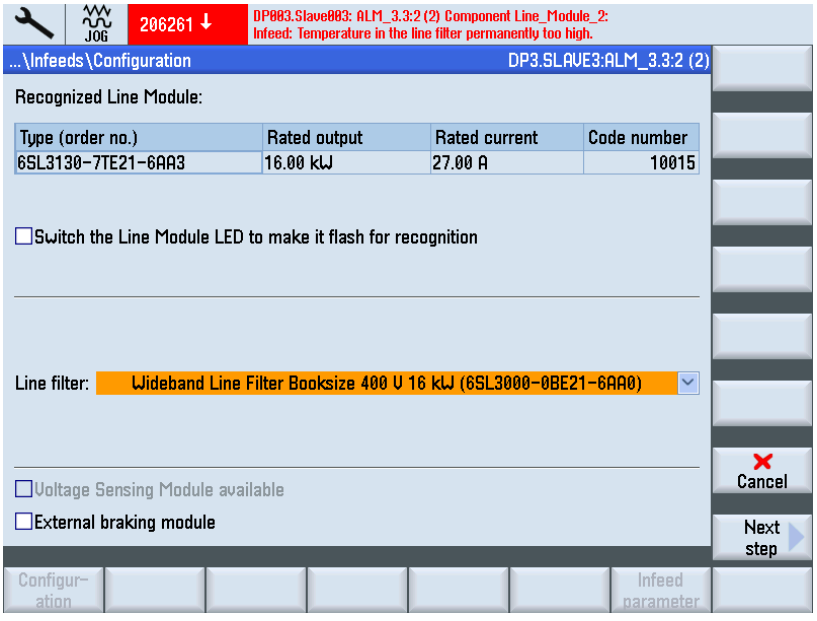



Figure 5-5 Infeeds > Configuration

The following functions can also be activated:

- If you activate the "LED to flash for recognition" option, then the selected infeed will flash alternately red-green.
- Select the filter type which is connected in front of the infeed from the list under "Line filter". If it is a filter with temperature monitoring (AIM = Active Interface Module), then this is automatically activated with the selection (default setting). This option must be activated if you have inserted a Basic Line Filter in front of the AIM on account of loss of the interference emission and radio interference suppression.
- This option is selected if a Voltage Sensing Module has been detected.
- If you activate the external braking module option, the monitoring for the braking module is activated. This requires the appropriate terminal wiring (parameter p3866 to terminal X21.4 for S120 Booksize,) to have been established, otherwise Alarm 206900 is output.
- If you set the option "Activate external braking module", the function module for the braking module is activated. This can be used to monitor the module.

  To do so, set the corresponding terminal wiring (e.g. parameter p3866 to terminal X21.4 for S120 Booksize), otherwise alarm 206900 is output. In addition, make a BICO connection from the infeed to the free terminals on the NCU. The parameters that must be connected for this purpose can be found in the function diagram for the braking module in the SINAMICS S120 List Manual.

3. Select "Next step >".

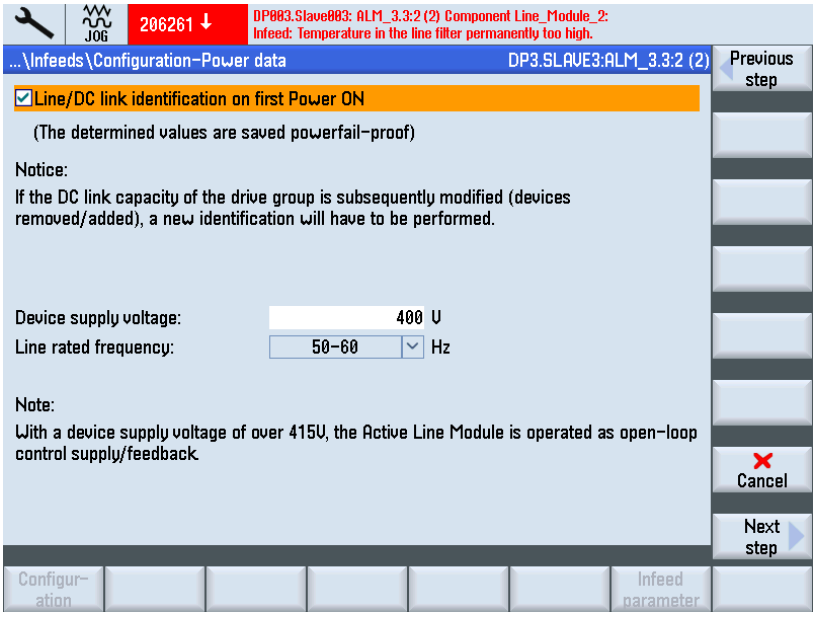



Figure 5-6 Infeeds > Line data

The following supply system data is configured:

– When selecting the checkbox, the line/intermediate circuit identification is activated once the pulse is enabled of the infeed (p3410). The infeed then switches to operational mode.

**Note**

**DC link identification**

If the line supply environment changes or the components on the intermediate circuit change (e.g. after setting up the equipment at the customer site or after expanding the drive group), the checkbox must be set again: Hence the "Supply system data" softkey in the overview in order to restart the supply system / DC link circuit identification.

If p3410 = 5 is saved in the commissioning archive, then the supply system / DC link identification starts automatically once the drive data is input into the archive.

Only then can it be guaranteed that the infeed operates with the optimum controller settings.

– Enter the device supply voltage: This is the basis for monitoring the supply voltage (p0281 - p0283), whereby an alarm is triggered in the case of an overshoot or undershoot (alarm threshold and shutdown threshold). The actual supply voltage is determined automatically and the adjustment is made based on this value.
– The actual line frequency for the infeed is determined automatically.
– In parameter p0284, p0285, set a threshold above which an alarm is triggered (default setting of the monitoring: 45 Hz to 65 Hz).

4. Select "Next step >".

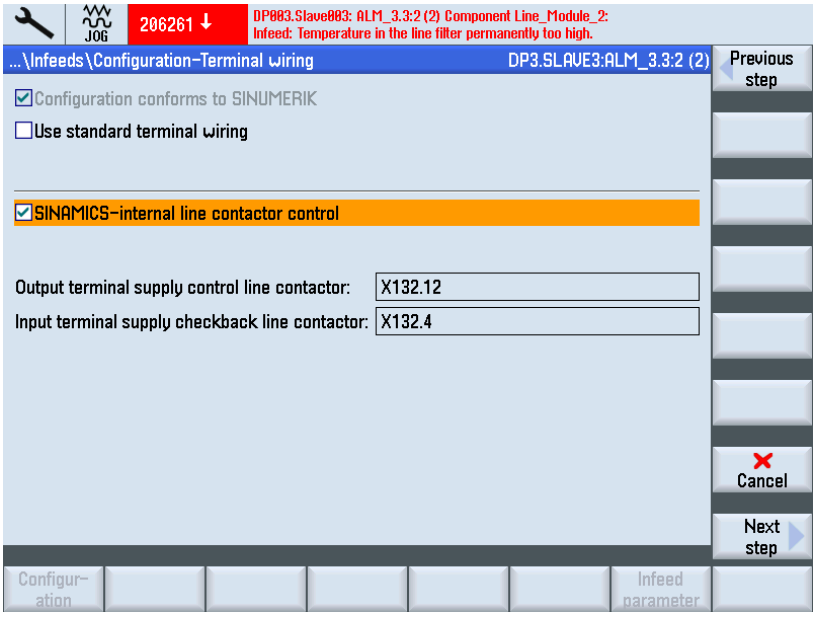



Figure 5-7 Infeeds > Terminal wiring

Use a line contactor to electrically isolate the drive line-up and the DC link from the line supply.

The checkbox for the infeed is selected by default. This means that the BICO interconnection (ON/OFF, both directions) is set with the first commissioning in accordance with the default setting. Activate the checkbox in order to restore the default setting for the terminal wiring.

If you use an external line contactor in front of the infeed, you can control it automatically via SINAMICS. To do this, set the checkbox to activate the "SINAMICS internal line contactor control" function.

Control via the PLC is also possible as an alternative. The connection is established using the displayed reserved terminals.

- **X132.12:** output terminal infeed to control the line contactor
- **X132.4:** input terminal infeed for line contactor feedback signal

It is not possible to internally control the line contactor using a drive object on the NCU, as the line contactor must be controlled from an external PLC or hardware control.

5. Select "Next step >". All data with which the infeed has been configured is displayed in the summary:

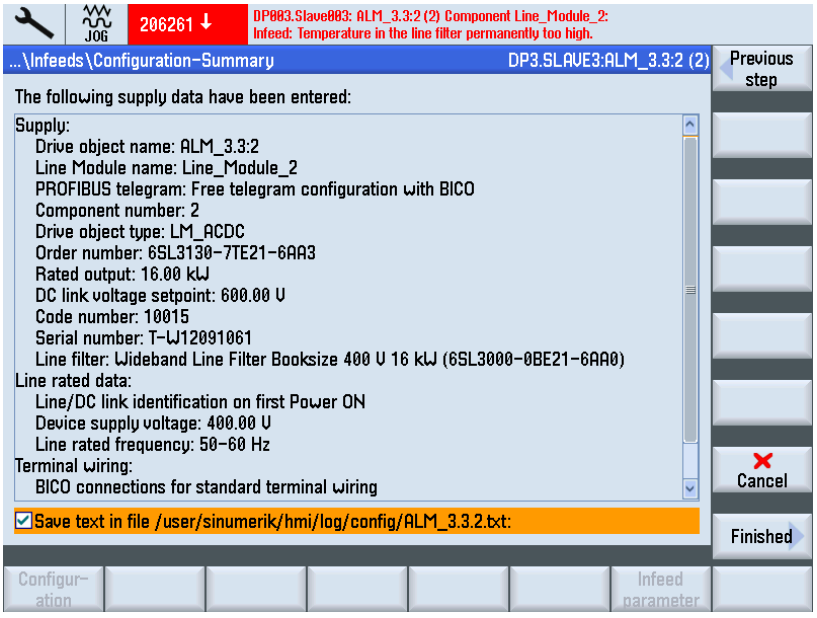



Figure 5-8 Infeeds > Summary

Further actions:

– Option: The summary can be saved in a text file.
– Press the "< Previous step" softkey to return to the previous view.
– "Finish" softkey to terminate the commissioning.

6. Select "Finish". In the event of changes, a save query is issued in each case: If you save the data, then it is retained upon Power On/Off. The data is saved on the CompactFlash card and can therefore be saved in a commissioning archive.

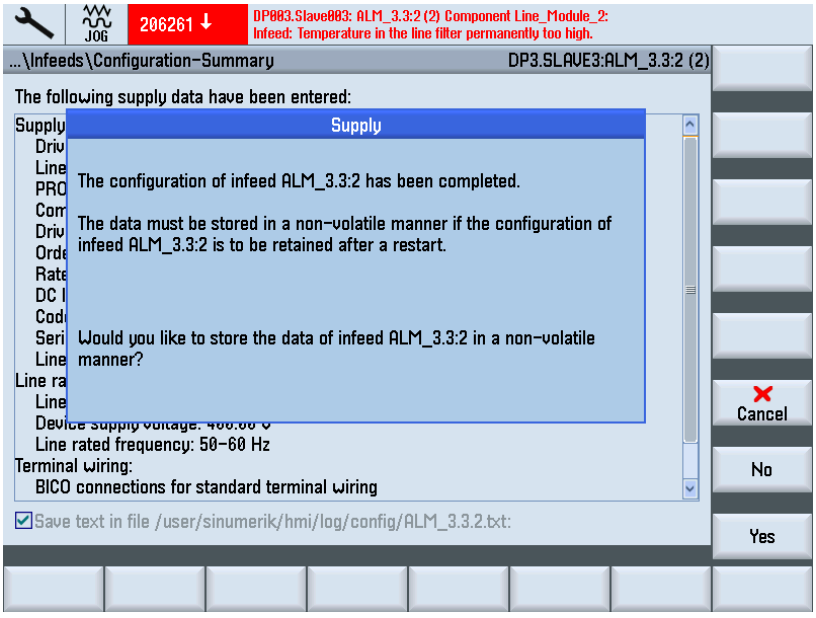



Figure 5-9 Saving the configuration

7. Select "Yes"

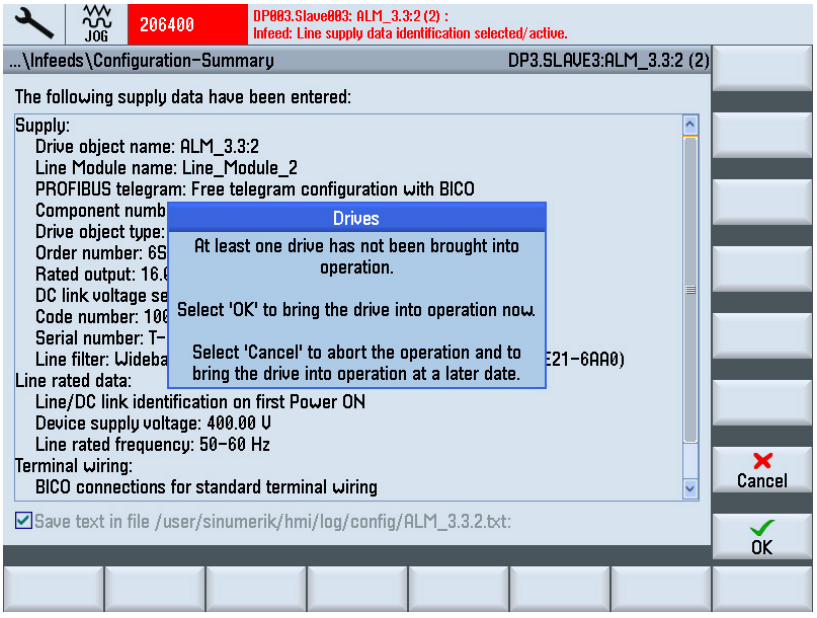



Figure 5-10 Drive screen

8. Select "Cancel" to check the overview of the infeed.

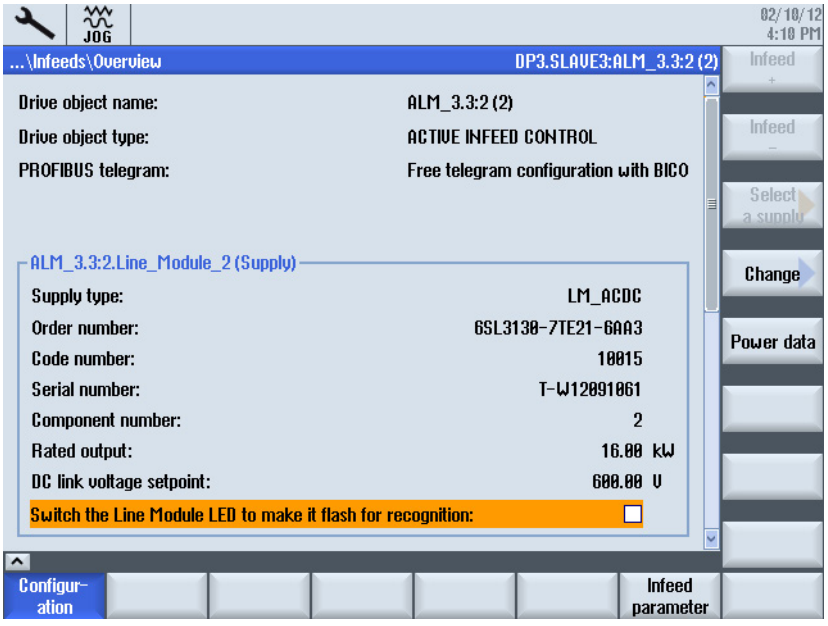



Figure 5-11 Overview of the infeeds (page1)

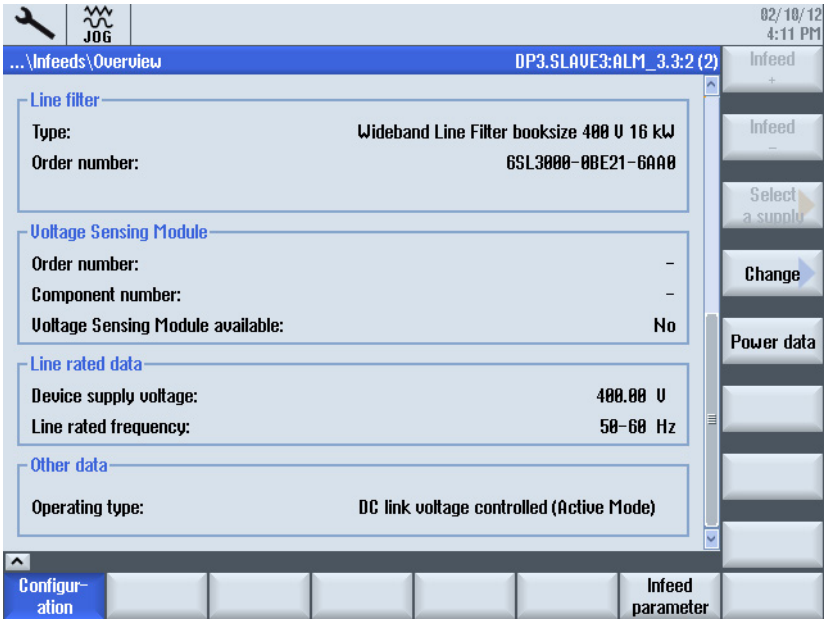



Figure 5-12 Overview of the infeeds (page2)

## 5.1.8 Replacing a SINAMICS S120 component

### Replacing a Motor Module

The replacement of components after the first commissioning, e.g. in order to use an ALM with a higher power 16 kW → 50 kW, must be performed in such a way that the configuration data is reimported and is up to date.

**Note**

**Firmware update**

The firmware of the configured drive components is updated correctly only when the components were inserted in the switched-off state. The subsequent insertion of drive components must **ONLY** be performed in the switched-off state.

If a drive component is replaced by a component with a different article number, e.g. a more powerful module, after first commissioning, observe the following sequence:

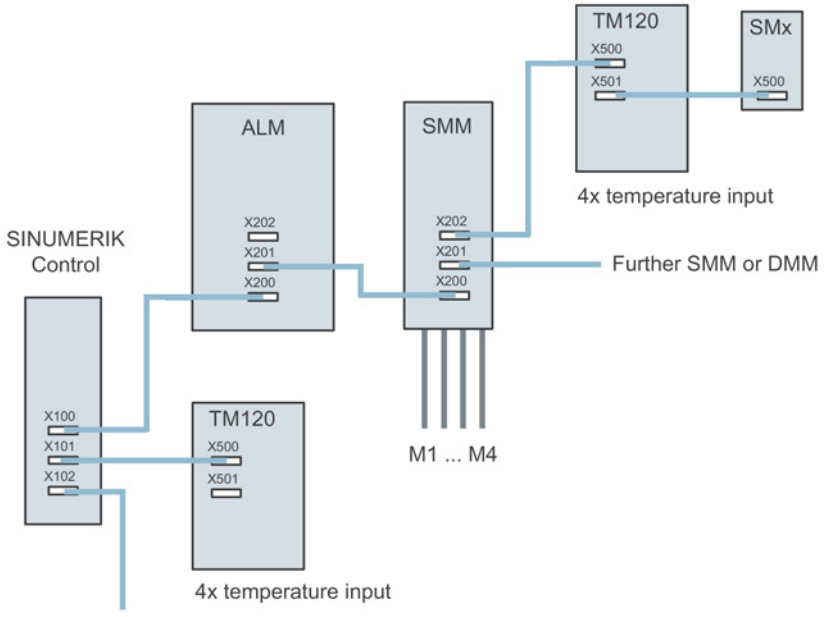
Procedure:

1. Select "Drive unit" → "Topology" → "Change".
2. Select the module and select "Deactivate drive object".
3. Replace the module in the cabinet.
4. Select the module and select "Activate drive object".
5. The configuration data of the new module is read in.
6. Update the drive data in the commissioning archive with the new configuration data.

## 5.1.9 Example: Parallel connection with TM120

### Topology

Application: Four motors connected in parallel




| | |
|---|---|
| M1 ... M4 | Motor 1 ... motor 4<br>One KTY and three PTC in series are connected for each motor. |
| SMx | Sensor Module (motor encoder) |
| ALM | Active Line Module |
| DMM | Double Motor Module |
| SMM | Single Motor Module |
| TM120 | Terminal Module |

The shown topology requires four KTY sensors and four PTC sensors:

- Each primary section has one KTY sensor (Temp-F) and three series-connected PTC sensors (Temp-S).

Two TM120s are required:

- A TM120 is interconnected automatically in series between the Motor Module and the SMx Sensor Module.
- A TM120 is directly inserted into a Line Module: This requires a manual interconnection by the commissioning engineer.

## Actions

Depending on the topology, the following actions must be performed on the TM120:

1. **TM120 between the Motor Module and the SMx Sensor Module**

   This TM120 evaluates four KTYs => sensor types must be selected by Servo-p4610/TM-p4100. The associated temperature is output by servo-r4620/TM120-r4105.

2. **TM120 directly on the Line Module**

   This TM120 evaluates four PTCs => sensor types must be selected by TM-p4100. The associated temperature is output by servo-r4105.

   Setting the threshold values in the PTC context:

   – TM120-p4102[x]=251 => evaluation Off

   – TM120-p4102[x]=120 => evaluation On

   Assignment of the TM120 response using fault propagation on the drive => setting of the propagation by servo-p0609=BICO:<object number>TM120:4105.0

## Assignment of the alarms to the motor

The occurring error messages are assigned to the motors as follows:

- Temperature channel on the TM120 in series connection with Motor Modules and encoders:

  **Alarm 207015** <location>drive: Motor temperature sensor warning

  **Alarm 207016** <location>drive: Motor temperature sensor fault

  **Alarm 235920** <location>TM: Temperature sensor channel 0 error

  In this case, the output message with reference to the motor and to the temperature component shows the relevant motor.

- Temperature channel on the TM120 directly to the Line Module:

  **Alarm 235207** <location>TM: Temperature fault/alarm threshold channel 0 exceeded

  Special feature with PTC:

  TM120-r4105 = -50 temperature is below the nominal response temperature

  TM120-r4105 = 250 temperature is above the nominal response temperature

  In this case, a message is output only with reference to the temperature component.

  The affected motor can be determined from the information concerning the TM120.

## 5.1.10 Exchanging process data

### Logical I/O address of drive

The I/O address informs the NC of the data area via which data is cyclically exchanged between the drive and the Control Unit:

- I/O address of the drive
- I/O address of the Control Unit
- Telegram type

The I/O addresses configured in the PLC are entered in the following machine data:

| | | |
|---|---|---|
| **Without NX module:** | | |
| MD13120[0] | $MN_CONTROL_UNIT_LOGIC_ADRESS | = 6500 |
| MD13120[1] | $MN_CONTROL_UNIT_LOGIC_ADRESS | = 0 |
| **With NX module:** | | |
| MD13120[0] | $MN_CONTROL_UNIT_LOGIC_ADRESS | = 6500 |
| MD13120[1] | $MN_CONTROL_UNIT_LOGIC_ADRESS | = 6516 |

### Telegram type

The default assignment of the drive telegram to type 136 is performed in the preconfigured system data block (SDB). During commissioning, this telegram type is set in the NC (MD13060 $MN_DRIVE_TELEGRAM_TYPE) and SINAMICS S120 ((p0922 PROFIdrive PZD telegram selection) components. The system data block is selected via MD11240 $MN_PROFIBUS_SDB_NUMBER **= n**.

### References

Further details on the process data can be found in:

- SINUMERIK 828D List Manual, "Machine Data"
- SINAMICS S120 Function Manual, Section "Communication"
- SINAMICS S120/S150 List Manual, at "Function diagrams"

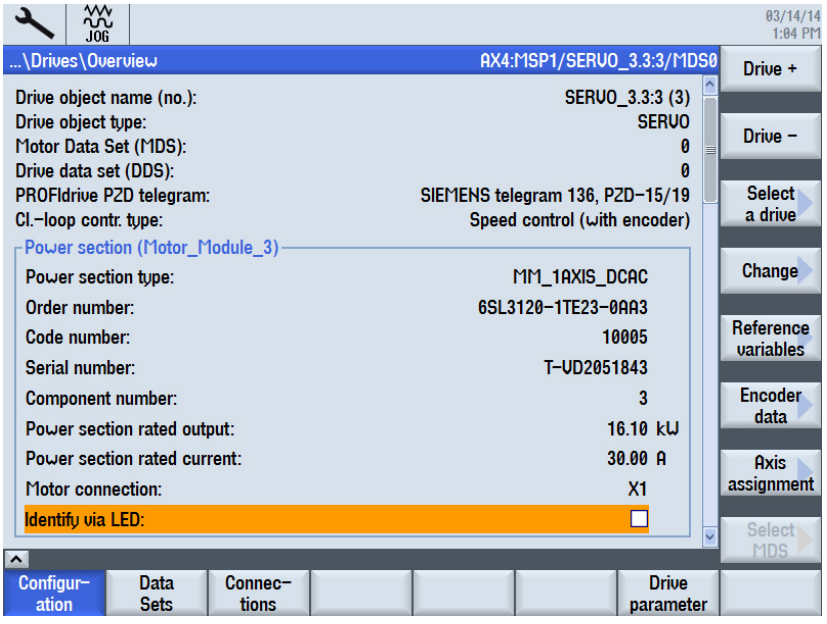
# 5.2 Assigning axes

## 5.2.1 Example: This is how you assign the axes

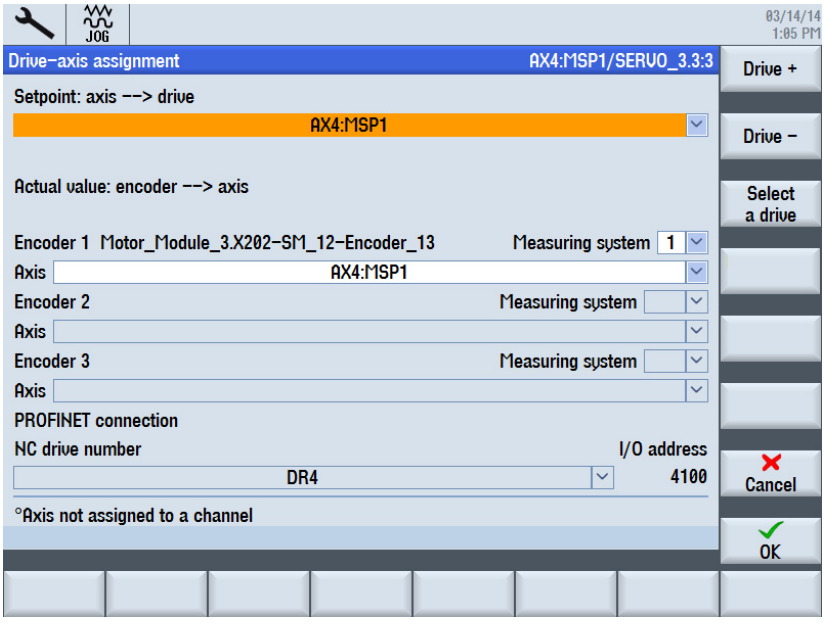
### Assigning axes

Procedure:

1. After you have finally saved the configuration data, change to the following overview:




2. To assign a real axis to every logical drive, select the "Axis assignment" softkey:



3. Mark the uppermost selection field for the "Setpoint: Axis → drive".
4. Select "AX1:MSP1" for the spindle from the selection list.
5. Mark the next selection field for the "Actual value: Encoder → axis" and assign a measuring system in accordance with the table below.
6. Select the next module with the "Drive +" and "Drive -" vertical softkeys to assign the other axes:

| Axis | Drive | I/O address |
|---|---|---|
| MSP1 | SERVO_3.3:3 | 4100 |
| MX1 | SERVO_3.3:4 | 4140 |
| MY1 | SERVO_3.3:5 | 4180 |
| MZ1 | SERVO_3.3:6 | 4220 |

7. Press "OK" to confirm the assignment and write the data to the NC. An NC reset is required for the data to take effect.

## Accepting the settings

After the restart, the following machine configuration is displayed on the control:

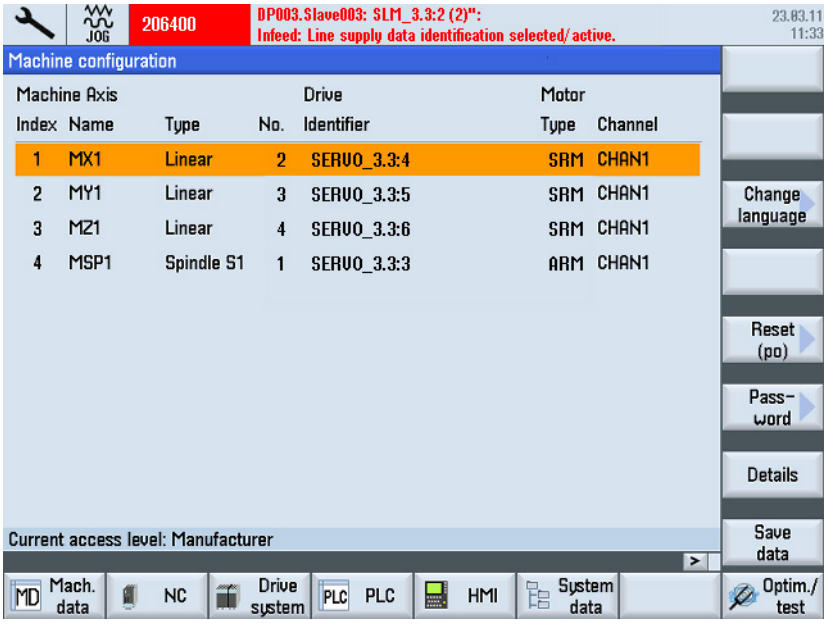



Figure 5-13 Machine configuration: All axes commissioned

Step 5 for commissioning a drive system has therefore been completed.

### Data backup

The configuration data is backed up after commissioning in the non-volatile memory with the "Save data" vertical softkey:

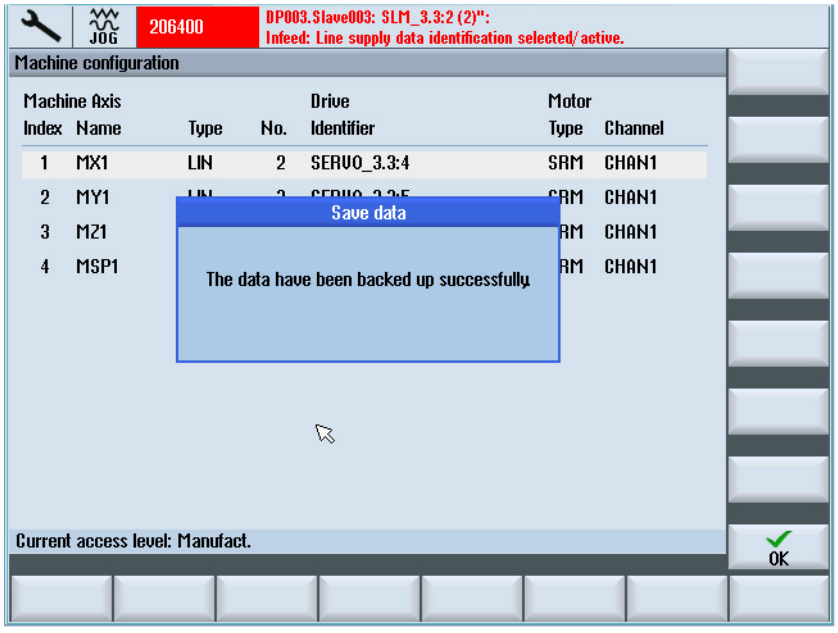



Figure 5-14 Message to back up data

## 5.2.2 Example: Setting machine data for an axis/spindle

### Axis machine data

After the commissioning in the previous example, the following values are set for the axis machine data:

| Axis machine data | | X | Y | Z | SP | A |
|---|---|---|---|---|---|---|
| MD30200 | $MA_NUM_ENCS | 1 | 1 | 1 | 1 | 1 |
| MD30230 | $MA_ENC_INPUT_NR | 1 | 1 | 1 | 2 | 1 |
| MD30240 | $MA_ENC_TYPE | 1 | 1 | 1 | 1 | 4 |
| MD31020 | $MA_ENC_RESOL | 2048 | 2048 | 2048 | 1024 | 512 |
| MD34200 | $MA_ENC_REFP_MODE | 1 | 1 | 1 | 1 | 0 |

## Spindle machine data

To operate the spindle in the previous example in JOG mode **after** commissioning the drive, enter the new values for the following machine data:

| Spindle machine data | | Default setting | New value |
|---|---|---|---|
| MD32000 | $MA_MAX_AX_VELO | 10000 | 3000 |
| MD32010 | $MA_JOG_VELO_RAPID | 10000 | 100 |
| MD32020 | $MA_JOG_VELO | 2000 | 50 |
| MD35100 | $MA_SPIND_VELO_LIMIT | 10000 | 3000 |
| MD35110[0] | $MA_GEAR_STEP_MAX_VELO[0] | 500 | 3000 |
| MD35110[1] | $MA_GEAR_STEP_MAX_VELO[1] | 500 | 3000 |
| MD35130[0] | $MA_GEAR_STEP_MAX_VELO_LIMIT[0] | 500 | 3150 |
| MD35130[1] | $MA_GEAR_STEP_MAX_VELO_LIMIT[1] | 500 | 3150 |
| MD36200[0] | $MA_AX_VELO_LIMIT[0] | 11500 | 3300 |
| MD36200[1] | $MA_AX_VELO_LIMIT[1] | 11500 | 3300 |

## Maximum speed drive parameter

After a DDS switchover, e.g. DDS0 → DDS1, the spindle can only be traversed with the speed from p1082[0]. If a value less than that in p1082[1 ... n] is entered in p1082[0], the spindle speed is limited to the value in p1082[0].

**Note**

The maximum speed must be entered in parameter p1082[0]:

- If the parameter is assigned a greater value than in the other parameters, the spindle can be traversed with the maximum speed.
- If parameter p1082[0] is subsequently changed to a lower value and then an NCK reset or Power OFF/ON performed, then the spindle only traverses with the lower value from p1082[0] and all other values are ignored.

## Diagnostics displays

Further information on the status of axes and spindle is available in the "Diagnostics" operating area with the menu forward key:

- The "Axis diagnostics" softkey opens the "Service Overview" dialog.
- The "Service axis" softkey opens the "Service Axis/Spindle" dialog.

**Note**

**DSC operation**

Requirement: Telegram ≥ 116; SINAMICS drive parameters:

- p1192[0] encoder selection
- p1193[0] encoder adaptation factor

The parameterization of the following combination is not permitted:
$MA_STIFFNESS_CONTROL_ENABLE=1 and $MA_ENC_FEEDBACK_POL=-1

The encoder inversion must be performed in the drive parameter p0410 bit 1 of the SINAMICS drive:

- DSC operation is preset for motor measuring systems.
- DSC operation must be explicitly activated for external measuring systems.

# 5.3 Commissioning auxiliary axes

## 5.3.1 Sample configuration – overview

### Connection of auxiliary axes via PROFINET

Auxiliary movements can be performed with auxiliary axes. They must not interpolate with other axes and therefore scarcely influence the performance capability of the system. Typical applications for example, are the tool and workpiece change.

You can connect up to two auxiliary axes on the SINUMERIK 828D via PROFINET. You require either a CU310-2 PN or CU320-2 PN for the connection of an auxiliary axis, see configuration examples below.

You must connect the cable for the PROFINET connection as follows:

- PN1 to the SINUMERIK 828D
- X150 P1 to the CU310-2 PN or CU320-2 PN



**Software option**

To be able to operate the auxiliary axes via the PLC I/O interface based on PROFINET, you require the "Positioning axis / auxiliary spindle" option.

## Sample configurations

The following configurations show examples with two auxiliary axes that are connected to the SINUMERIK 828D via the PLC I/O interface based on PROFINET:

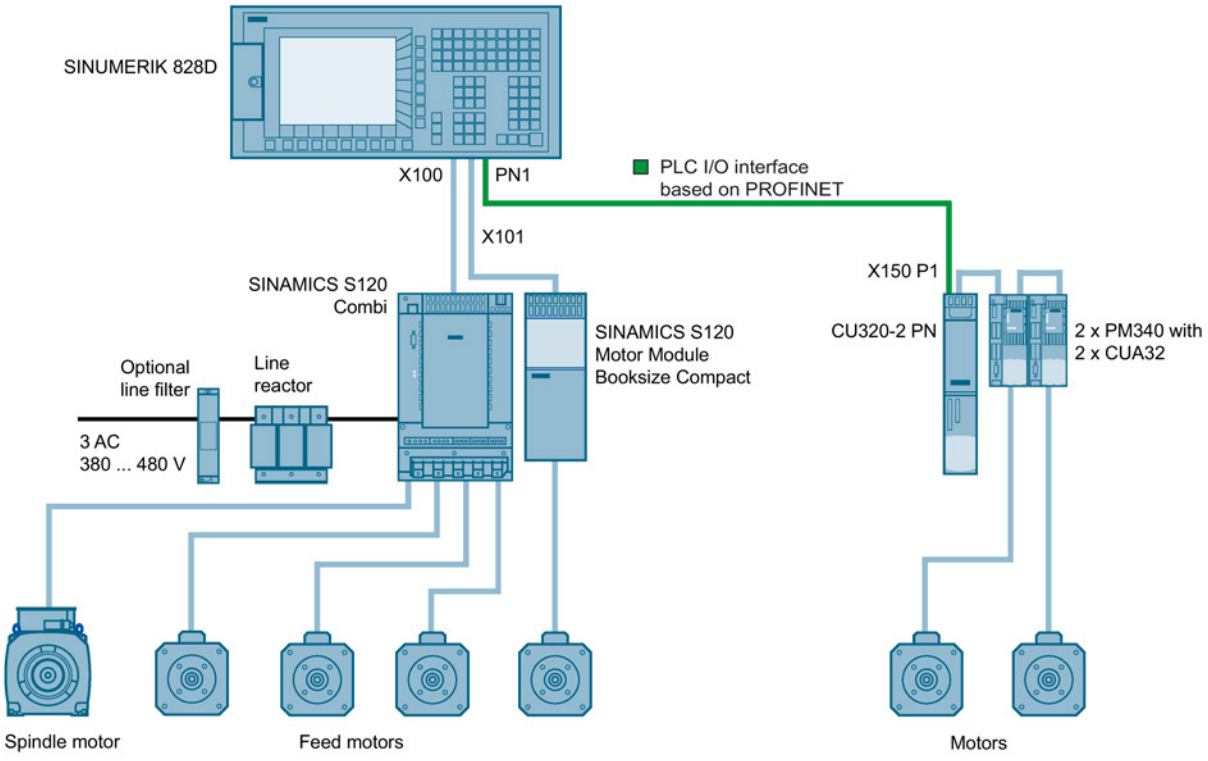



Figure 5-15 Configuration example 1: 2 auxiliary axes with 828D CU320-2 PN, 2 x PM340 and 2 x CUA32

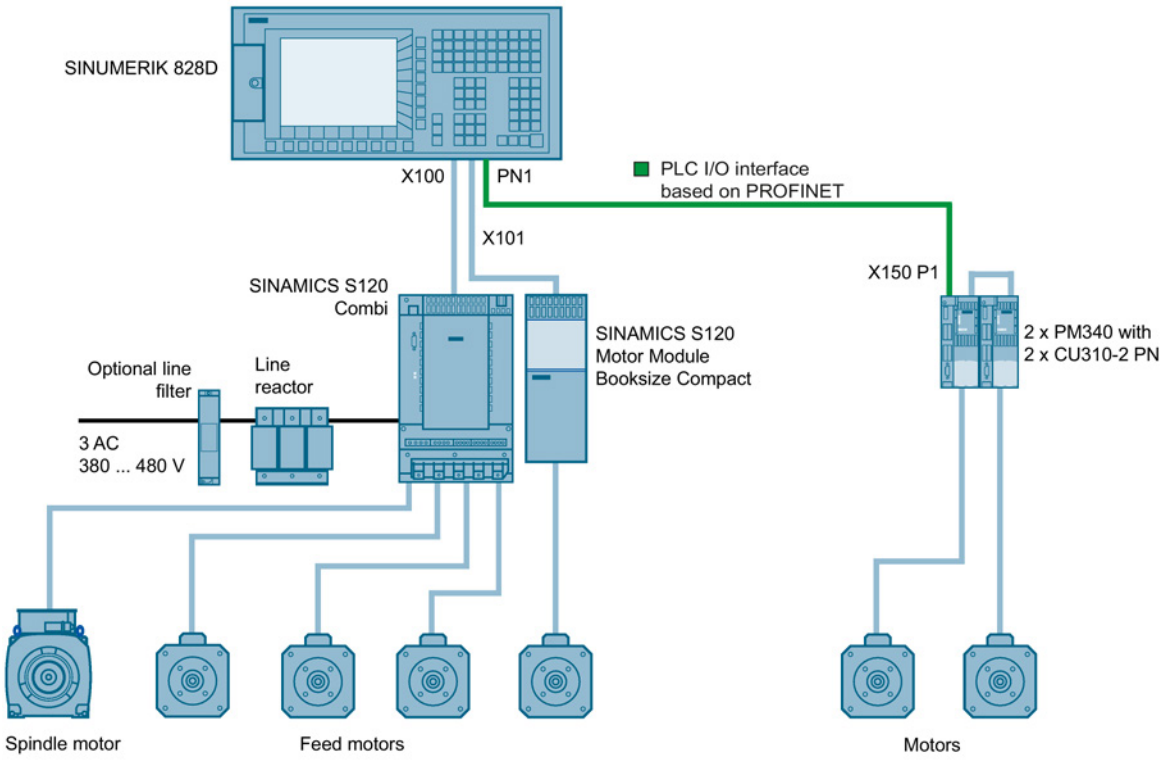



Figure 5-16 Configuration example 2: 2 auxiliary axes with 2 x CU310-2 PN and 2 x PM340

## 5.3.2 Example: How to configure an auxiliary axis

### Device name of the CU310-2 PN / CU320-2 PN

The device name of the CU310-2 PN / CU320-2 PN must match the name configured in the control. If the names do not match, the drive cannot be accessed by the control via PROFINET and cannot be commissioned.

Possible assignment of the name:

- If the CU310-2 PN / CU320-2 PN has not been initialized yet, it is automatically initialized by the control with the configured device name "MQ2".
- If the CU310-2 PN / CU320-2 PN has already been initialized, the device name must match the configured device name for fault-free operation.

## Activating the auxiliary axes

You can activate:

- The auxiliary axes "MQ2" and "MQ3" in the channel via MD20070 $MC_AXCONF_MACHAX_USED[].
- The auxiliary axis declaration for "MQ2" and "MQ3" via bit 8 in MD30460 $MA_BASE_FUNCTION_MASK.
- The appropriate SDB configuration via MD11240 $MN_PROFIBUS_SDB_NUMBER, see table below.

The changes take effect with a "Power on reset".

| Control Unit | SDB number | Module number | Axis name | IP address |
|---|---|---|---|---|
| 1x CU320-2 PN | (2,-1,2,2) | 1st axis = 30<br>2nd axis = 31 | 1st axis = MQ2<br>2nd axis = MQ3 | 192.168.214.19 |
| 1x CU310-2 PN | (4,-1,4,4) | 1st axis = 30 | 1st axis = MQ2 | 192.168.214.19 |
| 2x CU310-2 PN | (5,-1,5,5) | 1st axis = 30<br>2nd axis = 31 | 1st axis = MQ2<br>2nd axis = MQ3 | CU1: 192.168.214.19<br>CU2: 192.168.214.18 |

## Commissioning the drive

The commissioning and axis assignment described here refers to the first configuration example "2 auxiliary axes with CU320-2 PN, 2 x PM340 and 2 x CUA32". The procedure for the second configuration example is similar.

When the CU320-2 PN Control Unit can be accessed by the SINUMERIK 828D via PROFINET, you can commission the drive.

The connected auxiliary axes are detected and configured during the automatic device configuration, see Automatic first commissioning (Page 99).

To check/change the configuration of the drives:

1. Open the "Drives" window by selecting "Commissioning → Drive system → Drives".
2. Select "Change" to configure the drive.
3. Save the configuration data after a change.

## Performing the drive/axis assignment

After commissioning the drive, perform the drive/axis assignment.

Procedure:

1. Start the drive/axis assignment via "Commissioning → Drive system → Drives → Axis assignment".
2. Assign axis "MQ2" to drive "DR30" with I/O address "5260".

   The I/O address "5260" is in the machine data MD13050 $MN_DRIVE_LOGIC_ADDRESS [29].

| Axis | Drive | I/O address |
|---|---|---|
| MQ2 | DR30 | 5260 |
| MQ3 | DR31 | 5300 |

3. Assign axis "MQ3" to drive "DR31" with I/O address "5300".

   The I/O address "5300" is in the machine data MD13050 $MN_DRIVE_LOGIC_ADDRESS [30].

The drive and the axis have been commissioned.

# 5.4 Topology rules for DRIVE-CLiQ

## 5.4.1 Topology rules for S120 Combi

### Topology rules for DRIVE-CLiQ

There are fixed DRIVE-CLiQ topology rules for the S120 Combi. These rules must be observed. If these rules are violated, then a corresponding alarm is displayed.

### Assigning the DRIVE-CLiQ interfaces

Table 5- 1 Assigning the DRIVE-CLiQ interfaces on the S120 Combi

| DRIVE-CLiQ interface | Connection with |
|---|---|
| X200 | X100 of the PPU |
| X201 | Motor encoder, spindle |
| X202 | Motor encoder, feedrate 1 |
| X203 | Motor encoder, feedrate 2 |
| X204 | Motor encoder feed 3 → only for 4-axis S120 Combi<br>Remains empty for 3-axis S120 Combi |
| X205 | Optional: 2. direct sin/cos encoder for spindle (via SMx20) [1]<br>Remains empty when connecting a direct TTL spindle encoder via X220 |

[1] In this case, the TTL encoder interface X220 remains free

Table 5- 2 Assigning the DRIVE-CLiQ interfaces to the SINUMERIK 828D (PPU)

| DRIVE-CLiQ interface | Connection with |
|---|---|
| X100 | X200 of the S120 Combi |
| X101 | X200 of a Single Motor Module or Double Motor Module |
| X102 | X500 of the Terminal Module TM54F<br>X500 of the Hub Modules (DMx20) [1] |

[1] When using a TM54F, the DMx20 is connected in series at the TM54F via the DRIVE-CLiQ interface X501

Table 5- 3 Assigning the DRIVE-CLiQ interfaces of the expansion axes

| DRIVE-CLiQ interface | Connection with |
|---|---|
| First Single Motor Module | |
| X200 | X101 of the PPU |
| X201 [1)] | X200 of the second Single Motor Module |
| X202 | Motor encoder for feedrate 1st expansion axis (via Sensor Module) |
| Second Single Motor Module | |
| X200 | X201 of the first Single Motor Module |
| X201 | Remains empty |
| X202 | Motor encoder for feedrate 2nd expansion axis (via Sensor Module) |
| Double Motor Module | |
| X200 | X101 of the PPU |
| X201 | Remains empty |
| X202 | Motor encoder for feedrate 1st expansion axis |
| X203 | Motor encoder for feedrate 2nd expansion axis |

1) Remains empty if only one Single Motor Module is used

Table 5- 4 Assigning the DRIVE-CLiQ interfaces at the DMx20 to assign a direct measuring system to the feed axes.

| DRIVE-CLiQ interface | Feed axis |
|---|---|
| X500 | X501 of the TM54F<br>X102 of the PPU (if a TM54F is not being used) |
| X501 | Direct encoder, feedrate 1 at the S120 Combi |
| X502 | Direct encoder, feedrate 2 at the S120 Combi |
| X503 | Direct encoder, feedrate 3 at the S120 Combi → only for 4-axis S120 Combi<br>Remains empty for 3-axis S120 Combi |
| X504 | Feedrate 1st expansion axis at the Motor Module |
| X505 | Feedrate 2nd expansion axis at the Motor Module |

Table 5- 5 Assigning the DRIVE-CLiQ interfaces at the TM54F

| DRIVE-CLiQ interface | |
|---|---|
| X500 | X102 of the control (PPU) |
| X501 | X500 of the DMx20<br>If a DMx20 is not being used, then this interface remains empty. |

## 5.4.2 Topology rules for S120 Booksize

### Introduction

The following rules apply for wiring components with DRIVE-CLiQ. A distinction is made between **DRIVE-CLiQ rules** which must always be observed, and **recommended rules** which, when observed, do not require any subsequent changes to the topology when expansions are made.

The maximum number of DRIVE-CLiQ components and the possible wiring form depend on the following points:

- The binding DRIVE-CLiQ wiring rules
- The number and type of activated drives and functions on the respective Control Unit
- The computing power of the respective Control Unit
- The set processing and communication cycles

In addition to the binding wiring rules that must be observed, some additional recommendations as well as topology examples for DRIVE-CLiQ wiring are provided in the PPU manual.

The components used in these examples can be removed, replaced with others or supplemented. If components are replaced by another type or additional components are added, the SIZER tool should be used to check the topology.

**Note**

Every topology that SIZER permits can also run and is therefore correct (SINAMICS S120 D Function Manual /FH1/).

### DRIVE-CLiQ rules

The wiring rules below apply to standard cycle times (servo 125 µs). For cycle times that are shorter than the corresponding standard cycle times, additional restrictions apply due to the computing power of the Control Unit.

The rules below apply on a general basis, unless limited, as a function of the firmware version.

- A maximum of eight DRIVE-CLiQ nodes can be connected in one row. A row is always seen from the perspective of the Control Unit.

  This number reduces to a maximum of four DRIVE-CLiQ nodes in one row when the extended functions of drive-based Safety Integrated are configured.
- A maximum of 14 nodes can be connected to one DRIVE-CLiQ line on a Control Unit.

- Ring wiring is not permitted.
- Components must not be double-wired.

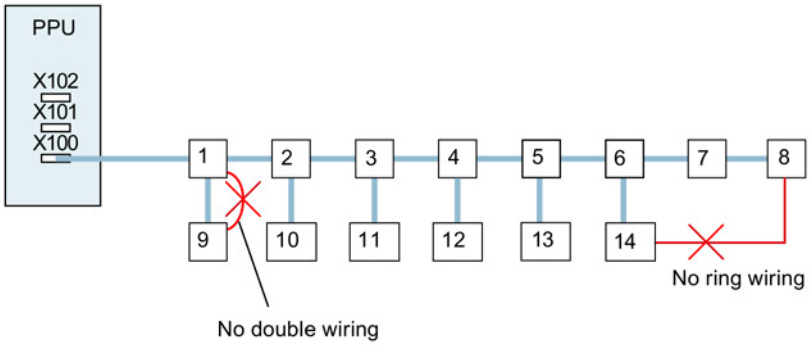



Figure 5-17 Example: DRIVE-CLiQ line at the X100 terminal (without Safety Integrated)

- The following applies to the booksize format:
  - In the servo control and vector U/f control operating modes, only one Line Module may be connected to the Control Unit. In the vector control operating mode, a maximum of three further Line Modules may be connected in parallel (i.e. a total of four Line Modules).
  - It is permissible that one Line Module and Motor Modules are connected together to one DRIVE-CLiQ line in the servo control mode.
  - One Line Module and Motor Modules must be connected to separate DRIVE-CLiQ lines in the vector control mode.
  - For booksize format, a parallel connection of Infeed Modules or Motor Modules is not possible.
- The following applies to the chassis format: Line Modules (Active Line, Basic Line, Smart Line) and Motor Modules must be connected to separate DRIVE-CLiQ lines.
- The default sampling times may be changed.
- An NX10.3 / NX15.3 must be connected to the PPU at X102.
- A maximum of 16/20 measuring systems can be connected to a PPU with NX10.3/NX15.3:

  **Example 1:** PPU with six axes with six motor measuring systems and six direct measuring systems as well as NX10.3 with two motor measuring systems and two direct measuring systems.

  **Example 2:** PPU with five axes with five motor measuring systems and five direct measuring systems as well as NX10.3 with three motor measuring systems and three direct measuring systems.

  **Example 3:** PPU with six axes with six motor measuring systems and six direct measuring systems as well as NX15.3 with four motor measuring systems and four direct measuring systems.
- Only one TM54F is permitted for each PPU and for each NX10.3/NX15.3.

- The Active Line Module Booksize and the Motor Modules Booksize can be connected to one DRIVE-CLiQ line.
- Chassis Line Module and Motor Module are connected in series.
- To allow the following modules to be assigned automatically during the commissioning (device identification), they should be connected to a free DRIVE-CLiQ port on the associated Active Line Module / Motor Module:
  - Voltage Sensing Module (VSM)
  - Terminal Module TM120

  **Note**

  If the TM120 is connected without observing this rule, the commissioning engineer must use BICO technology to assign the temperature channels to the drive.

- The sampling times (p0115[0] and p4099) of all components that are connected to a DRIVE-CLiQ line must be divisible by one another with an integer result. If the current controller sampling time on a DO has to be changed to another pattern that does not match the other DOs on the DRIVE-CLiQ line, the following options are available:
  - Reconnect the DO to a separate DRIVE-CLiQ line.
  - Also change the current controller sampling time and the sampling time of the inputs/outputs of the DO not involved so that they again fit into the time grid.

  **Note**

  A Double Motor Module, a DMC20, and a TM54F each correspond to two DRIVE-CLiQ nodes. This also applies to Double Motor Modules, of which just one drive is configured.

To enable the function "Automatic configuration" to assign the encoders to the drives, the recommended rules below must be observed.

Recommended rules:

- The DRIVE-CLiQ cable from the Control Unit must be connected as follows:
  - To X200 of the first booksize power unit
  - To X400 of the first chassis power unit
- The DRIVE-CLiQ connections between the power units must each be connected from interface X201 to X200 or from X401 to X400 on the follow-on component.

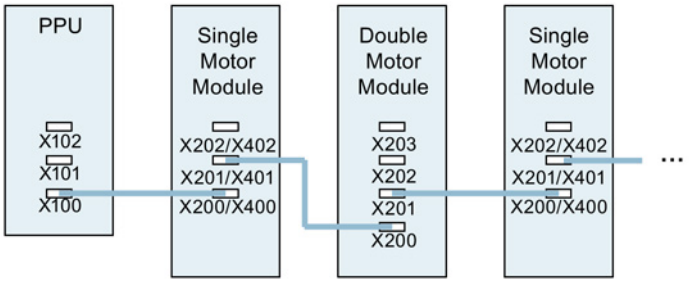



Figure 5-18 Example: DRIVE-CLiQ line

- The motor encoder must be connected to the associated power unit.

| Component | Connecting the motor encoder via DRIVE-CLiQ |
|---|---|
| Single Motor Module booksize | X202 |
| Double Motor Module booksize | • Motor connection X1: Encoder at X202<br>• Motor connection X2: Encoder at X203 |
| Single Motor Module chassis | X402 |
| Power Module chassis | X402 |

**Note**

If an additional encoder is connected to a Motor Module, it is assigned to this drive as encoder 2 in the automatic configuration. At a Double Motor Module, an encoder at X201 is assigned to the 2nd feedrate as 2nd measuring system.

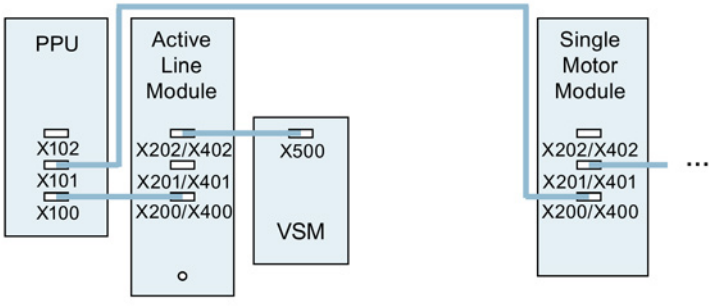



Figure 5-19 Example: Topology with VSM for booksize and chassis components

| Component | VSM connection |
|---|---|
| Active Line Module booksize | X202 |
| Active Line Module (chassis) | X402 |
| Power Modules | The VSM is not supported. |

## 5.4.3 Topology rules for SMC40

### Use of the SMC40

The SMC40 Sensor Module Cabinet-Mounted is used to convert encoder signals from absolute encoders with EnDat 2.2 to DRIVE-CLiQ. Two encoder systems with EnDat 2.2 can be connected to the SMC40. Their signals are converted independent of each other on two DRIVE-CLiQ encoder signals.

### Connection conditions

To ensure that the SMC40 Sensor Module Cabinet-Mounted is integrated into the topology during the initial commissioning, you must absolutely observe the following rules:

- Connect at least one of the DRIVE-CLiQ interfaces X500/1 or X500/2 on the SMC40 using DRIVE-CLiQ.
- Connect one EnDat encoder to the corresponding encoder interface X520/1 (to X500/1) or X520/2 (to X500/2).
- Operate the SMC40 only in a star topology. The DRIVE-CLiQ X500/1 and X500/2 sockets cannot be used for a series connection.

| NOTICE |
| --- |
| **Display in the "Topology" dialog**<br>The SMC40 is only integrated into the actual topology if the DRIVE-CLiQ interfaces X500/x and the corresponding encoder interfaces X520/x are assigned.<br>Without a connected encoder, it is also not possible to subsequently integrate the SMC40 in the topology. |

# 5.5 Terminal assignments

## 5.5.1 Terminal assignment at X122 and X132

### X122: Digital inputs/outputs of the PPU

| Pin | | Signal name | Status | Meaning |
|---|---|---|---|---|
| 1 | DI0 | DI0 | r722.0 | Digital input 0 |
| 2 | DI1 | DI1 | r722.1 | Digital input 1 |
| 3 | DI2 | DI2 | r722.2 | Digital input 2 |
| 4 | DI3 | DI3 | r722.3 | Digital input 3 |
| 5 | DI16 | DI16 | r722.16 | Digital input 16 |
| 6 | DI17 | DI17 | r722.17 | Digital input 17 |
| 7 | M2 | MEXT2 | --- | Ground for pins 1...6 |
| 8 | P1 | P24EXT1 | --- | +24 V power supply |
| 9 | IO8 | DI/DO8 | r722.8 / r747.8 | Digital input/output 8 |
| 10 | IO9 | DI/DO9 | r722.9 / r747.9 | Digital input/output 9 |
| 11 | M1 | MEXT1 | --- | Ground for pins 9, 10, 12, 13 |
| 12 | IO10 | DI/DO10 | r722.10 / r747.10 | Digital input/output 10 |
| 13 | IO11 | DI/DO11 | r722.11 / r747.11 | Digital input/output 11 |
| 14 | M1 | MEXT1 | --- | Ground for pins 9, 10, 12, 13 |

### X132: Digital inputs/outputs of the PPU

| Pin | | Signal name | Status | Meaning |
|---|---|---|---|---|
| 1 | DI4 | DI4 | r722.4 | Digital input 4 |
| 2 | DI5 | DI5 | r722.5 | Digital input 5 |
| 3 | DI6 | DI6 | r722.6 | Digital input 6 |
| 4 | DI7 | DI7 | r722.7 | Digital input 7 |
| 5 | DI20 | DI20 | r722.20 | Digital input 20 |
| 6 | DI21 | DI21 | r722.21 | Digital input 21 |
| 7 | M2 | MEXT2 | --- | Ground for pins 1...6 |
| 8 | P1 | P24EXT1 | --- | +24 V power supply |
| 9 | IO12 | DI/DO12 | r722.12 / r747.12 | Digital input/output 12 |
| 10 | IO13 | DI/DO13 | r722.13 / r747.12 | Digital input/output 13 |
| 11 | M1 | MEXT1 | --- | Ground for pins 9, 10, 12, 13 |
| 12 | IO14 | DI/DO14 | r722.14 / r747.14 | Digital input/output 14 |
| 13 | IO15 | DI/DO15 | r722.15 / r747.15 | Digital input/output 15 |
| 14 | M1 | MEXT1 | --- | Ground for pins 9, 10, 12, 13 |

## 5.5.2 Terminal assignment at X242 and X252

### X242 pin assignment

| Pin | | Signal name | NC variable | Meaning |
|---|---|---|---|---|
| 1 | Not connected | | | |
| 2 | Not connected | | | |
| 3 | IN1 | DIN1 | $A_IN[1] | Digital NC input 1 |
| 4 | IN2 | DIN2 | $A_IN[2] | Digital NC input 2 |
| 5 | IN3 | DIN3 | $A_IN[3] | Digital NC input 3 |
| 6 | IN4 | DIN4 | $A_IN[4] | Digital NC input 4 |
| 7 | M4 | MEXT4 | | Ground for pins 3...6 |
| 8 | P3 | P24EXT3 | | +24 V power supply |
| 9 | O1 | DOUT1 | $A_OUT[1] | Digital NC output 1 |
| 10 | O2 | DOUT2 | $A_OUT[2] | Digital NC output 2 |
| 11 | M3 | MEXT3 | | Ground for pins 9, 10, 12, 13 |
| 12 | O3 | DOUT3 | $A_OUT[3] | Digital NC output 3 |
| 13 | O4 | DOUT4 | $A_OUT[4] | Digital NC output 4 |
| 14 | M3 | MEXT3 | | Ground for pins 9, 10, 12, 13 |

### X252 pin assignment

| Pin | | Signal name | NC variable | Meaning |
|---|---|---|---|---|
| 1 | AO | AOUT | | Analog output (voltage for analog spindle) |
| 2 | AM | AGND | | Analog ground |
| 3 | IN9 | DIN9 | $A_IN[9] | Digital NC input 9 |
| 4 | IN10 | DIN10 | $A_IN[10] | Digital NC input 10 |
| 5 | IN11 | DIN11 | $A_IN[11] | Digital NC input 11 |
| 6 | IN12 | DIN12 | $A_IN[12] | Digital NC input 12 |
| 7 | M4 | MEXT4 | | Ground for pins 3...6 |
| 8 | P3 | P24EXT3 | | +24 V power supply |
| 9 | O9 | DOUT9 | $A_OUT[9] | Digital NC output 9 |
| 10 | O10 | DOUT10 | $A_OUT[10] | Digital NC output 10 |
| 11 | M3 | MEXT3 | | Ground for pins 9, 10, 12, 13 |
| 12 | O11 | DOUT11 | -- | • Without analog spindle:<br>Digital NC output 11<br>• With analog spindle:<br>Controller release for analog spindle (in accordance with MD30134 $MA_IS_UNIPOLAR_OUTPUT) |

| Pin | | Signal name | NC variable | Meaning |
|---|---|---|---|---|
| 13 | O12 | DOUT12 | -- | • Without analog spindle:<br>Digital NC output 12<br>• With analog spindle:<br>Travel direction of the analog spindle (in accordance with MD30134 $MA_IS_UNIPOLAR_OUTPUT) |
| 14 | M3 | MEXT3 | | Ground for pins 9, 10, 12, 13 |

## See also

Machine data for an analog spindle (Page 81)

## 5.5.3 Example: Circuitry for a CU with line contactor

### Example

The circuitry shown refers to the assignment of the terminals in the previous chapters.

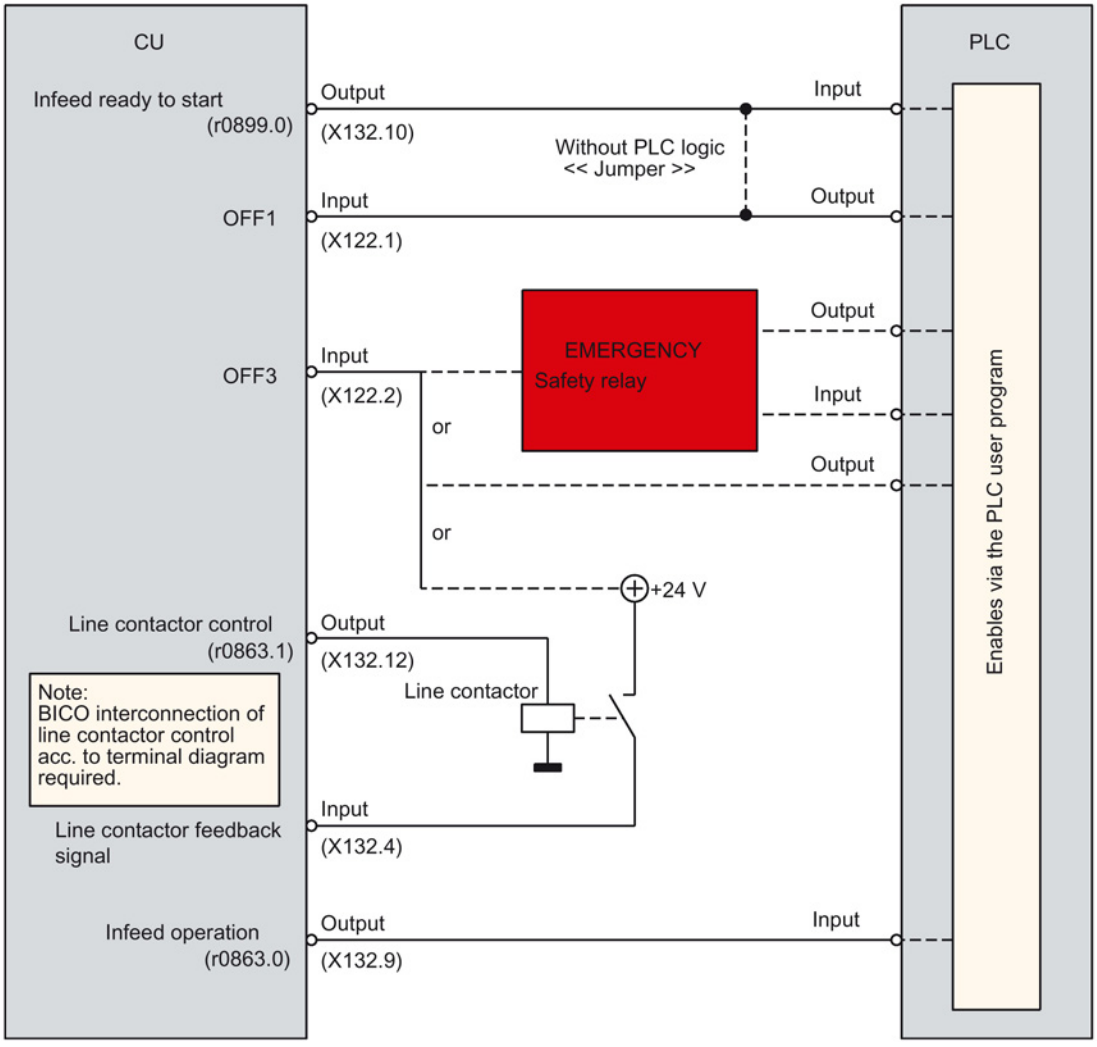
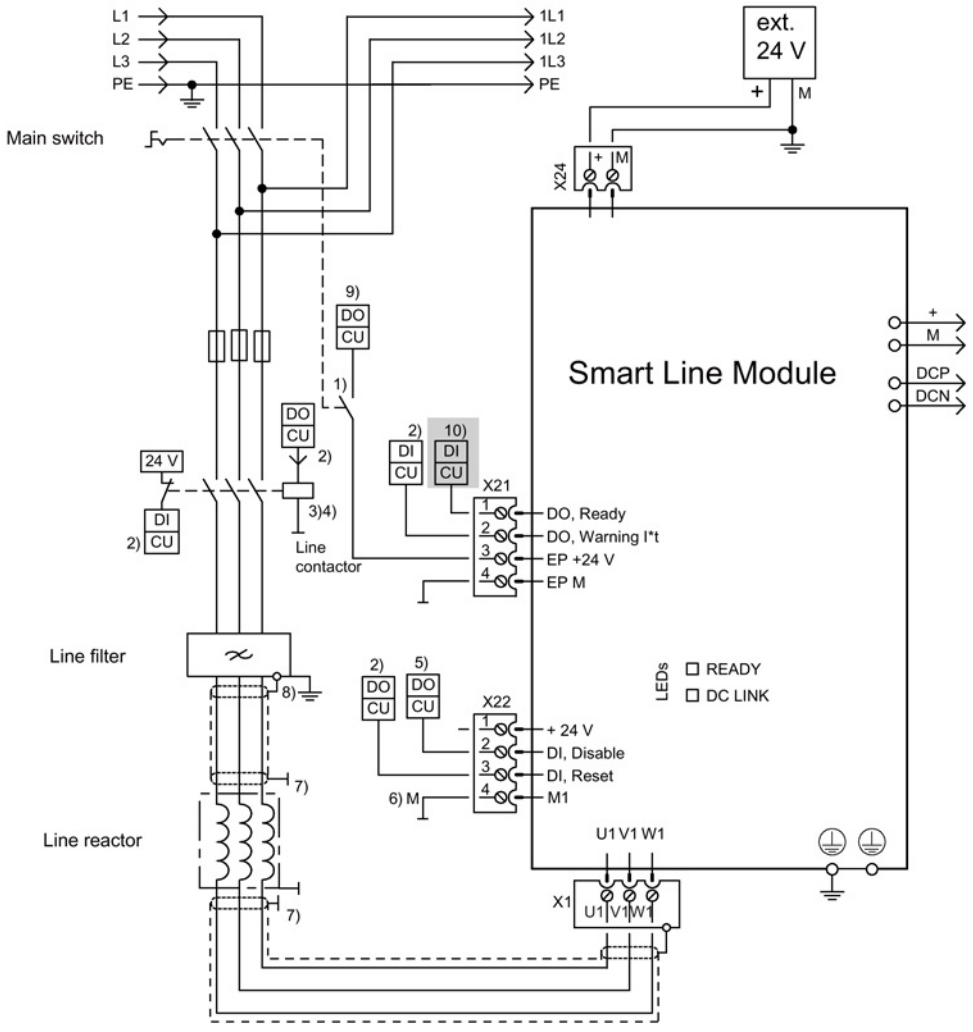



Figure 5-20 Circuitry for a Control Unit with line contactor

### Smart Line Module connection

The connections for the digital in/outputs X122 and X132 are on the rear side of the control.




① Early opening contact t > 10 ms, 24 VDC and ground must be applied for operation
② DI/DO controlled by the Control Unit
③ No additional load permitted downstream of line contactor
④ The current carrying capacity of the DO must be observed; an output interface must be used if required.
⑤ DO high, feedback deactivated (a jumper can be inserted between X22 pin 1 and pin 2 for permanent deactivation).
⑥ X22 pin 4 must be connected to ground (external 24 V).
⑦ Contacting via rear mounting panel or shielding buses in accordance with EMC installation guideline
⑧ 5 kW and 10 kW line filters via shield connection
⑨ Signal output of the control, to avoid interference of the 24 VDC supply on the EP terminal.
⑩ **Connect via BICO to parameter p0864 → X122.1**

Figure 5-21 Example: SLM connection

ON/OFF1 enable: Connection of Smart Line Module pin X21.1 → X122.1 SINUMERIK 828D

Connect further input and output signals to the PLC I/O:

- DI → PLC inputs
- DO → PLC outputs

### See also

Additional information can be found in:

- SINUMERIK 828D Manual PPU
- SINAMICS S120 Manual Booksize Power Units

## 5.5.4 Inputs/outputs (dialog)

### Support for the terminal assignment

The following overview shows the terminal assignment of those drive units involved in the SINAMICS drive line-up.

"Commissioning" → "Drive system" → "Drive units" → "Inputs/outputs" operating area:

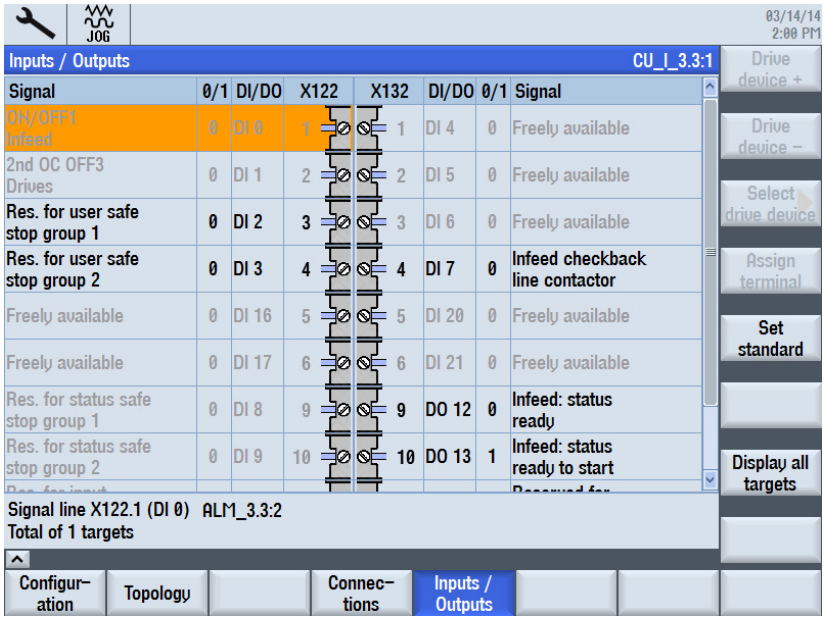



Figure 5-22 Terminal assignment of the inputs/outputs

## Axis/spindle test run

**Drive:**

| Parameter/terminal | Meaning |
|---|---|
| p0840 | ON/OFF1 |
| p0844 | 1. OFF2 |
| p0845 | 2. OFF2 |
| p0848 | 1. OFF3 |
| p0849 | 2. OFF3 |
| p0852 | Enable operation |
| X21.3 (+24 V) and X21.4 (ground) | EP terminals enable (pulse enable) |
| p0864 | Infeed enable |
| p1140 | Ramp-function generator enable |
| p1141 | Ramp-function generator start |
| p1142 | Setpoint enable |

**Infeed:**

| Parameter/terminal | Meaning |
|---|---|
| p0840 | ON/OFF1 |
| p0844 | 1. OFF2 |
| p0845 | 2. OFF2 |
| p0852 | Enable operation |
| X21.3 (+24 V) and X21.4 (ground) | EP terminals enable (pulse enable) |

## See also

Further references on the drive:

- SINAMICS S120 Commissioning Manual
- Manual Booksize power units

## 5.5.5 Interconnections (dialog)

### Interconnections

Each drive unit contains a large number of interconnectable input and output variables as well as internal control variables. The drive unit can be adapted to a wide range of requirements using BICO technology (Binector Connector Technology).

Digital and analog signals, which can be interconnected as required by means of BICO parameters, are identified by the prefix BI, BO, CI, or CO in their parameter name. These parameters are identified accordingly in the parameter list or in the function diagrams:

- Binectors (digital): BI: Binector input, BO: Binector output
- Connectors (analog): CI: Connector input, CO: Connector output

### Display at the control

To interconnect two signals, an input parameter (signal sink) must be assigned to the required output parameter (signal source). The following dialog is available to check or to change the interconnection of the components involved in the SINAMICS drive line-up:

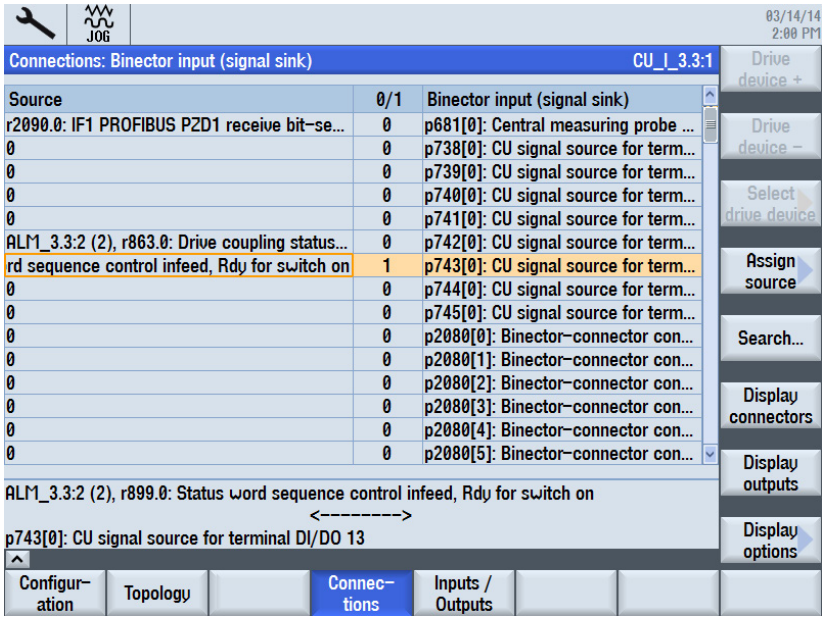



Figure 5-23 Example: Interconnections

## 5.5.6 Connecting the probes

### Connecting the probes

The probes are connected to both the SINUMERIK 828D CU and the NX:

1. Probe at terminal X122 pin 13 / X122 pin 13 of the NX
2. Probe at terminal X132 pin 13 / X122 pin 12 of the NX

**Note**

A precondition for measuring with the SINUMERIK 828D is that the distributed (local) measurement function has been parameterized.

Central measurement is not possible with SINUMERIK 828D.

### Machine data

The following machine data should be checked and adjusted if necessary:

- **General machine data:**

  MD13200[0] $MN_MEAS_PROBE_LOW_ACTIVE = 0 or 1

  MD13200[1] $MN_MEAS_PROBE_LOW_ACTIVE = 0 or 1

  Value 0 = deflected state 24 V (default)

  Value 1 = deflected state 0 V

  MD13210 $MN_MEAS_TYPE = 1 distributed measurement

- **Axis-specific machine data:**

  MD30244[0] $MA_ENC_MEAS_TYPE = 1 for all axes

  MD30244[1] $MA_ENC_MEAS_TYPE = 1 for all axes

  **Note**

  The machine data MD13210 and MD30244 are preset to the value 1 and cannot be changed! (Data class: S)

- **Control Unit parameters:**

  p0680[0] Central probe input terminal = 0

  p0680[1] Central probe input terminal = 0

  p0680[2] Central probe input terminal = 0

  The 1st probe is connected to terminal X122 pin 13, the 2nd probe to terminal X132 pin 13 of the SINUMERIK 828D and - if present - on the NX10 module to terminal X122 pin 12.

  A precondition is the switchover of pin X132.13 from output to input.

  Set CU input or output = set p0728 bit 15 to 0 (DI/DO X132.13)

- **Drive parameters:**

  p0488[0] Probe 1 input terminal: Encoder 1 = 3 → connector X122.13

  p0488[1] Probe 1 input terminal: Encoder 2 = 3 → connector X122.13

  p0488[2] Probe 1 input terminal: Encoder 3 = 0 → not used

  p0489[0] Probe 2 input terminal: Encoder 1 = 6 → connector X132.13

  p0489[1] probe 2 input terminal: Encoder 2 = 6 → connector X132.13

  p0489[2] Probe 2 input terminal: Encoder 3 = 0 → not used

  The 2nd probe at terminal X122.12 must be parameterized for all axes that are parameterized on the NX module:

  p0489[0] Probe 2 input terminal: Encoder 1 = 6 → connector X122.12

  p0489[1] probe 2 input terminal: Encoder 2 = 6 → connector X122.12

  p0489[2] Probe 2 input terminal: Encoder 3 = 0 → not used

  **Note**

  All drives must be parameterized.

# 5.6 Configuring data sets

## 5.6.1 Data sets - overview

### Requirement

**Note**

The particular drives must have been already commissioned.

### Data sets

The data sets are configured in the "Commissioning" → Drive system" → "Drives" → "Data sets" operating area. The following prompts lead you through the sequences:

- "Add data set"
- "Remove data set" if an additional data set has already been created
- "Modify data set"

Number of configurable data sets:

- Motor data set → MDS0...3 (max. 4)
- Drive data set → DDS0...31 (max. 8 per MDS)
- Encoder data sets → EDS0...2 (max. 3)

**Note**

**Drive parameters that affect data sets**

Default setting p2038 = 0 for Interface Mode: SINAMICS

When a message frame is selected via p0922, the p2038 parameter affects the device-specific assignment of the bits in the control and status words.

p0922 = 100 ... 199 sets p2038 = 1 automatically and blocks the changing of p2038. This sets permanently the "SIMODRIVE 611U" Interface Mode with eight DDS per MDS for these message frames.

### Application

The Data Set Assistant (DSA) offers machine manufacturers (OEM) with a simple and efficient solution for duplicating and modifying data in order to configure the behavior of drives, motors and encoders. The reuse of the DSA data helps to minimize the configuration effort required to generate the machine data for matching the various parts.

Motor Data Sets (MDS), Drive Data Sets (DDS) and Encoder Data Sets (EDS) are used to configure the component characteristics of the drive system. The reuse of data reduces the configuration levels required for matching the machine dynamics of the individual parts.

### References

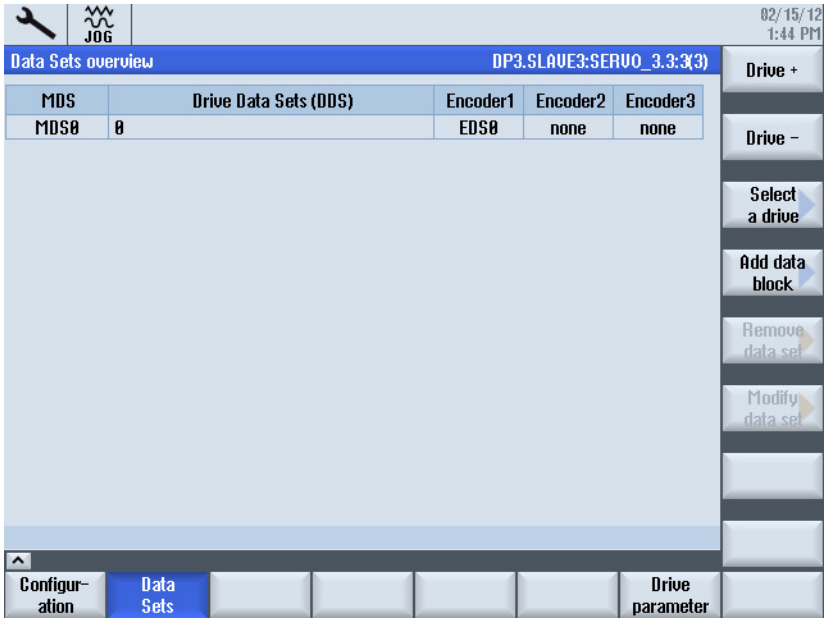
You can find additional information in:

SINUMERIK 840D sl/828D Basic Functions Function Manual, various NC/PLC interface signals and functions (A2)

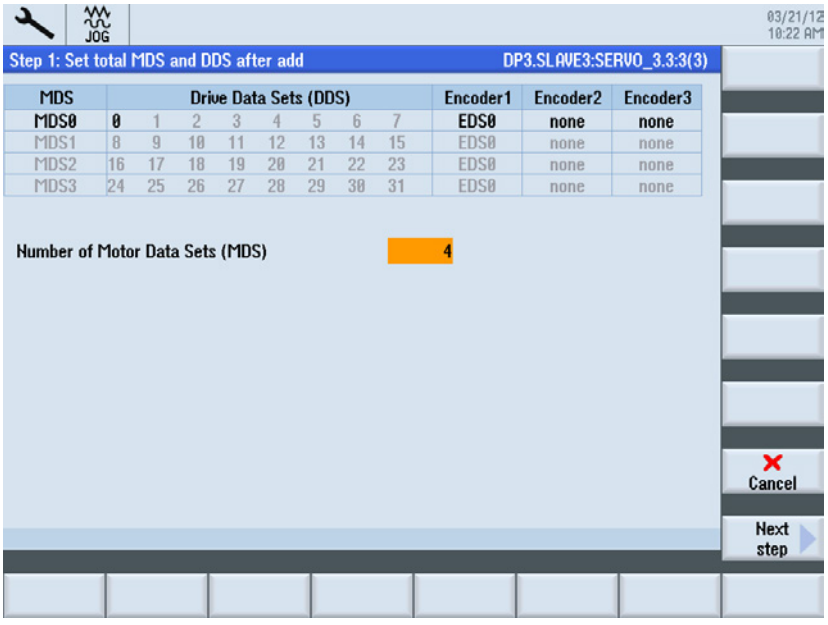
## 5.6.2 Adding a data set

### Adding a data set operating sequence

The default factory setting specifies an MDS0 motor data set with a DDS0 drive data set and an EDS0 encoder data set:

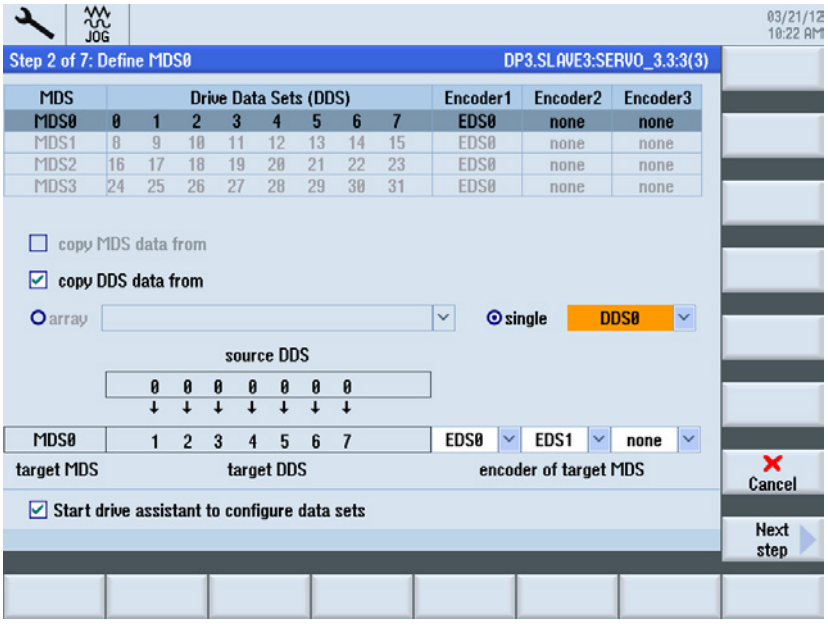


1. Press "Add data set". In the example, the maximum number of four MDSes are created.

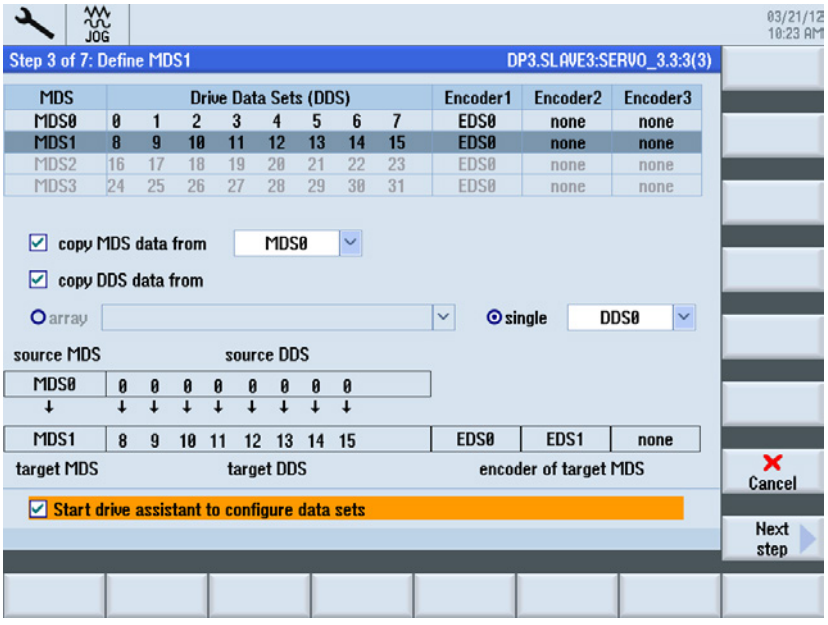



2. Press "Next step >".

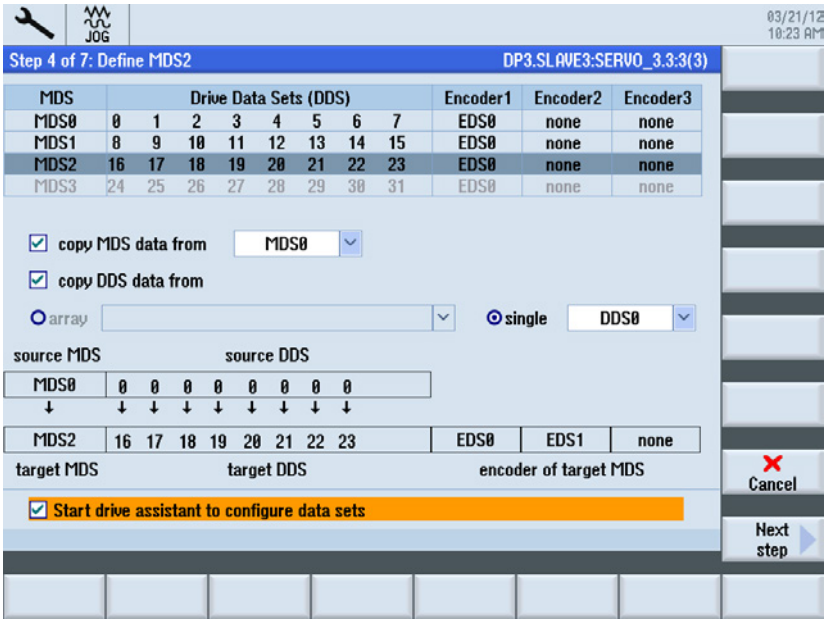
   In the first step, the DDS1 to DDS7 drive data sets are created in the MDS0.



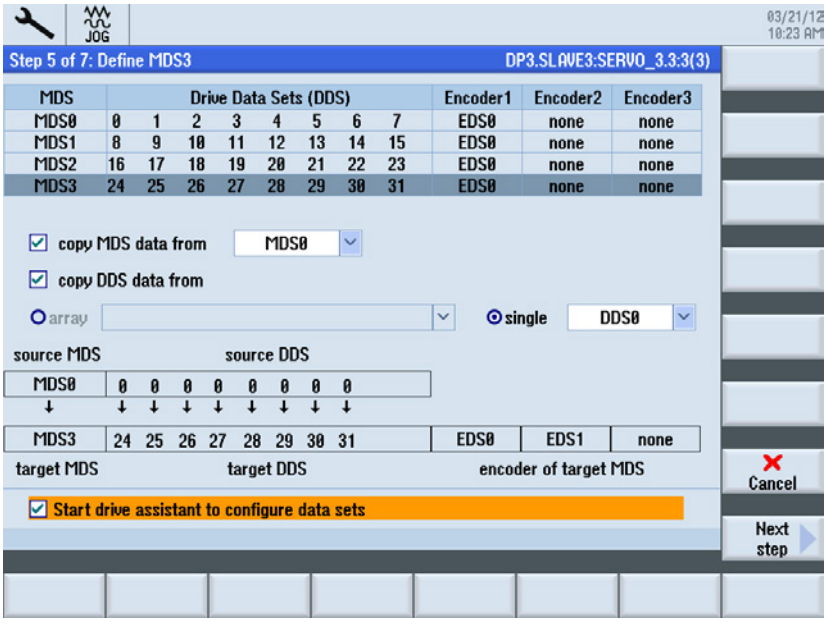
3. Press "Next step >". The MDS1 motor data set is created.



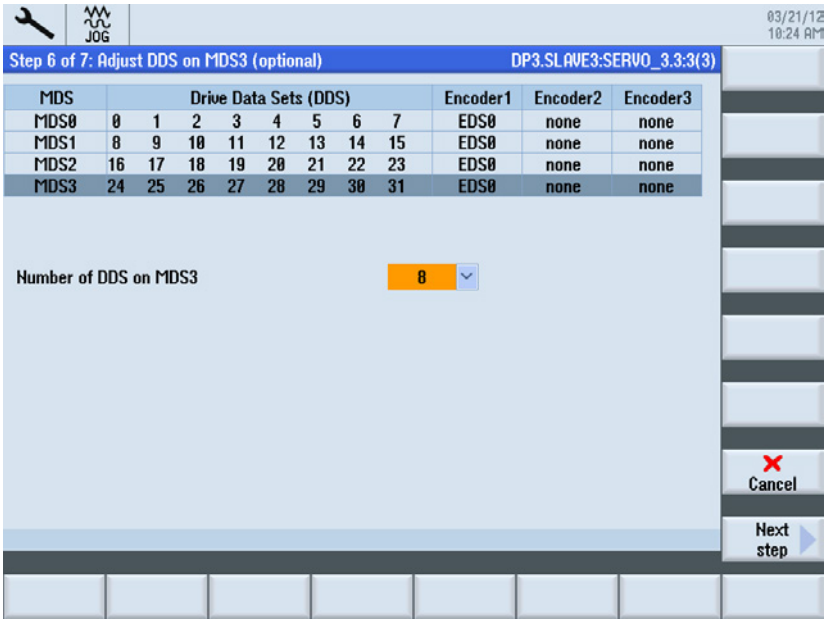

4. Press "Next step >". The MDS2 motor data set is created.



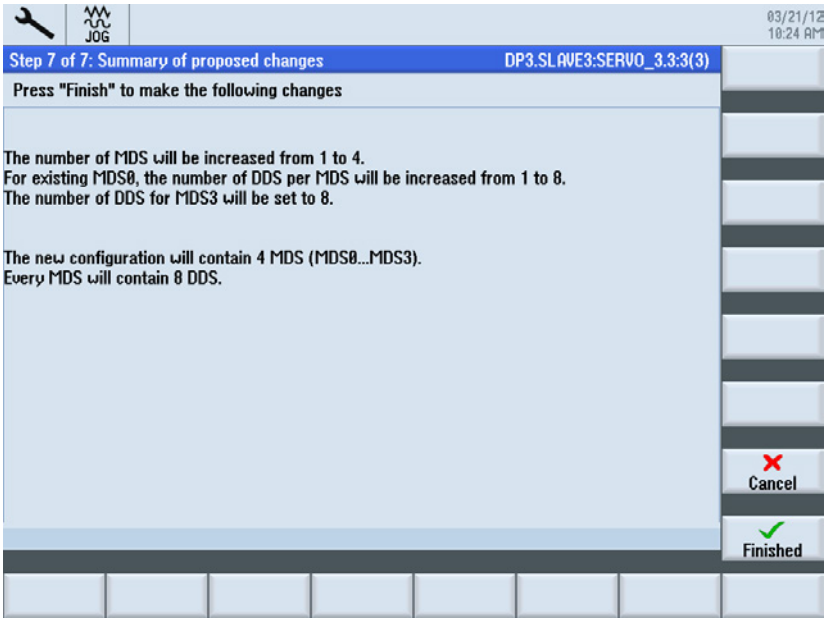
5. Press "Next step >". The MDS3 motor data set is created.



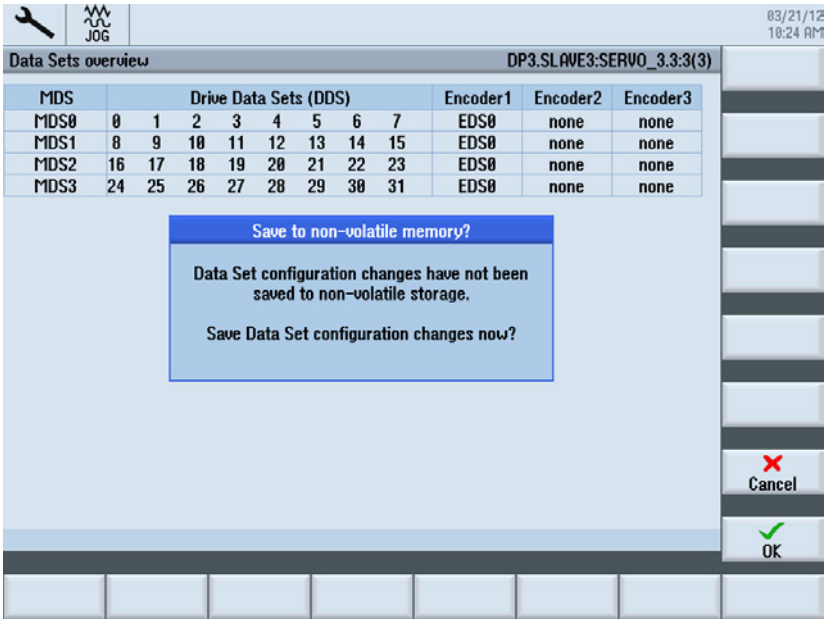

6. Press "Next step >". You need to create a complete MDS as shown in the following diagram, or as an option, enter the number of DDS for the MDSx:



7. Press "Next step >" to check in the summary the changes to be performed.



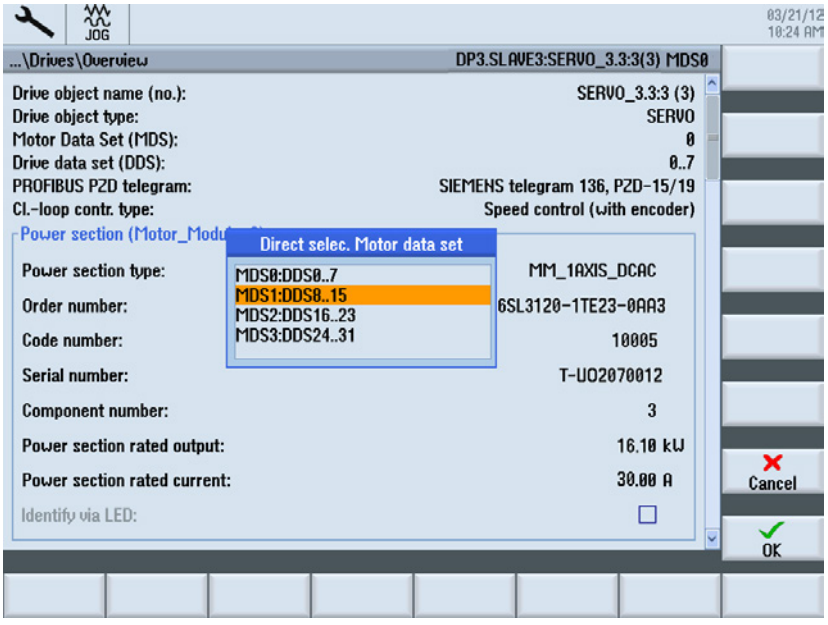

8. Press "Finish >" to accept the changes.




9. Confirm with "OK" to save the data in non-volatile memory.

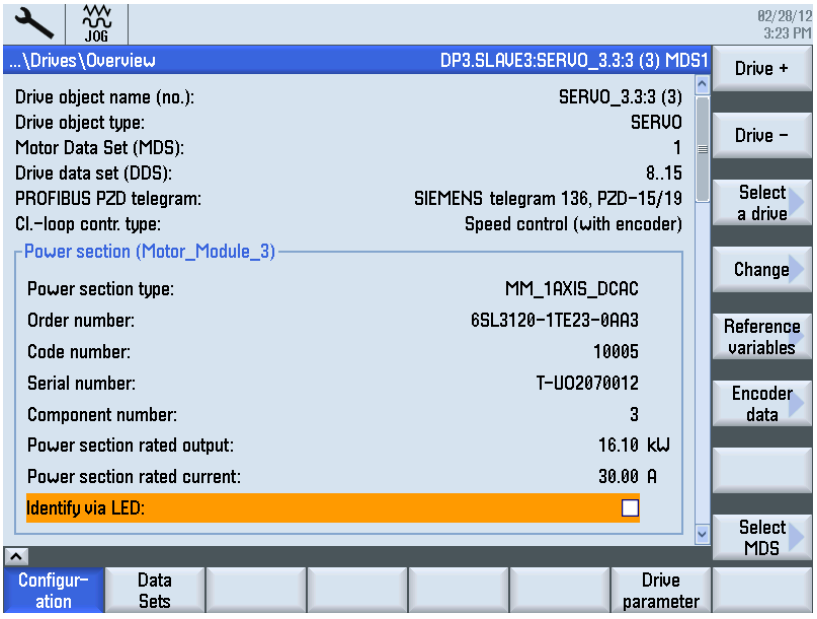
   The saving/writing of the parameters can take several minutes.

10. The "Drives" → "Overview" dialog of the drive object is displayed. The "Select MDS >" softkey is now available. Press "Select MDS...".




In the direct selection, select the motor data set, e.g. in this case MDS1.

11. Press "OK" to display the new motor data set for the drive object.




The drive is now operational.

## 5.6.3 Remove data set

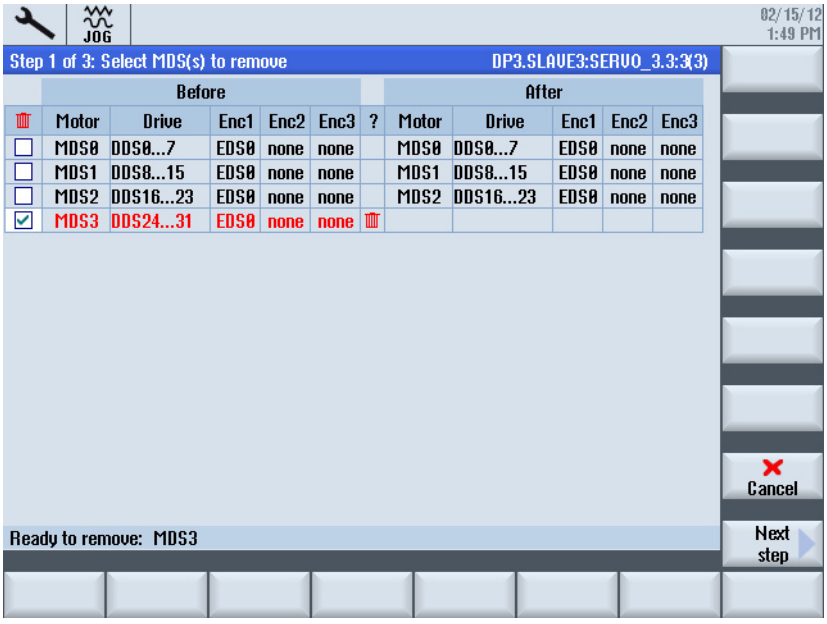
### Precondition

To remove data sets, the following conditions must be met:

- Number of DDS > 1 in the MDS ⇒ DDSes can be removed.
- Number of MDS > 1 ⇒ MDSes can be removed.

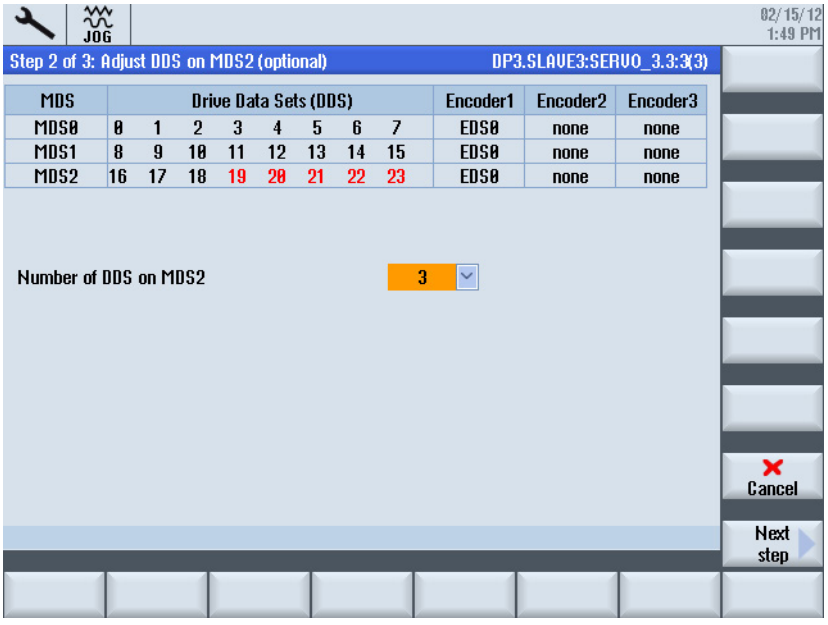
### Operating sequence to remove a data set

1. Select "Remove data set":



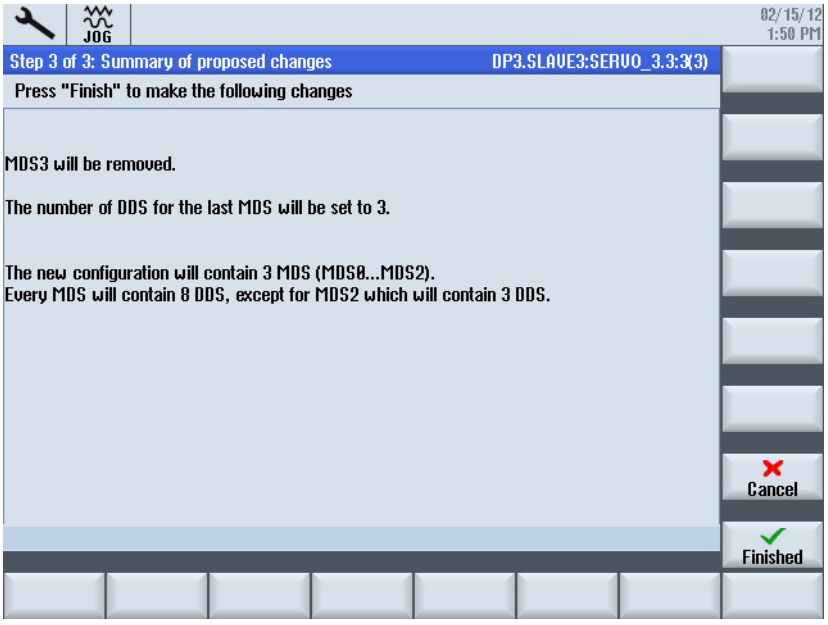

In the first column, select the MDS to be removed by marking the row with a check mark. Several data sets can also be selected.

2. Press "Next step >".




As an option, you can remove the remaining DDSes from MDS2 by entering the number of DDS drive data sets that should remain in the MDS: By entering 3 DDS in MDS, five DDSes are removed.

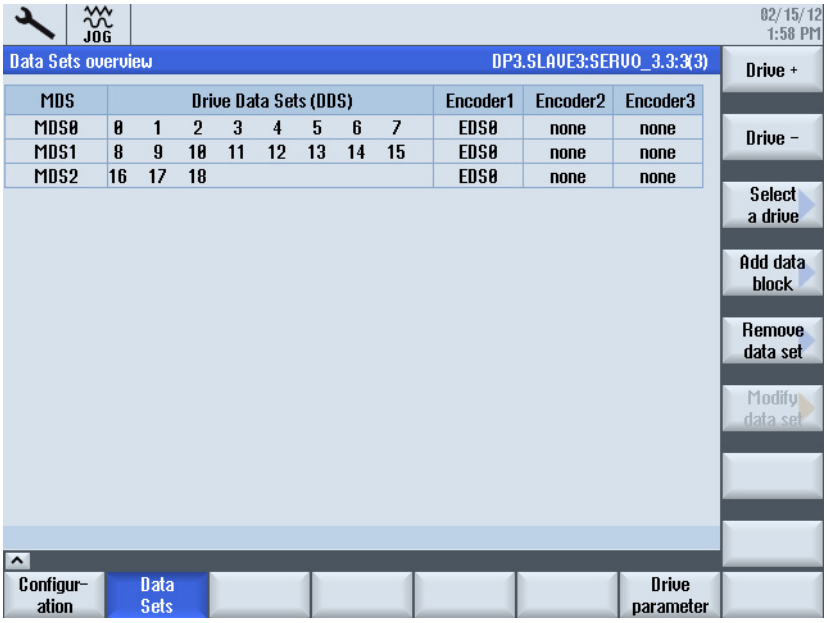
3. Press "Next step >".




"Cancel" terminates this process and the data set overview with MDS0 ... MDS3 is displayed.

4. Press "Finish >" to accept the changes.

   The data is written to the parameters. The results are then displayed:




When you exit this dialog, press "Yes" to confirm the prompt to save the data to the non-volatile memory.

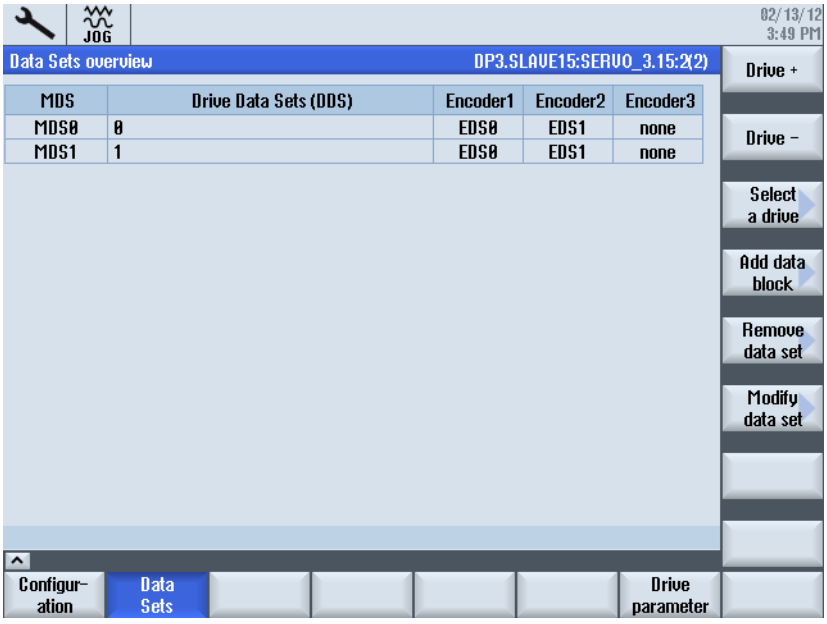
## 5.6.4 Modify data set

### Precondition

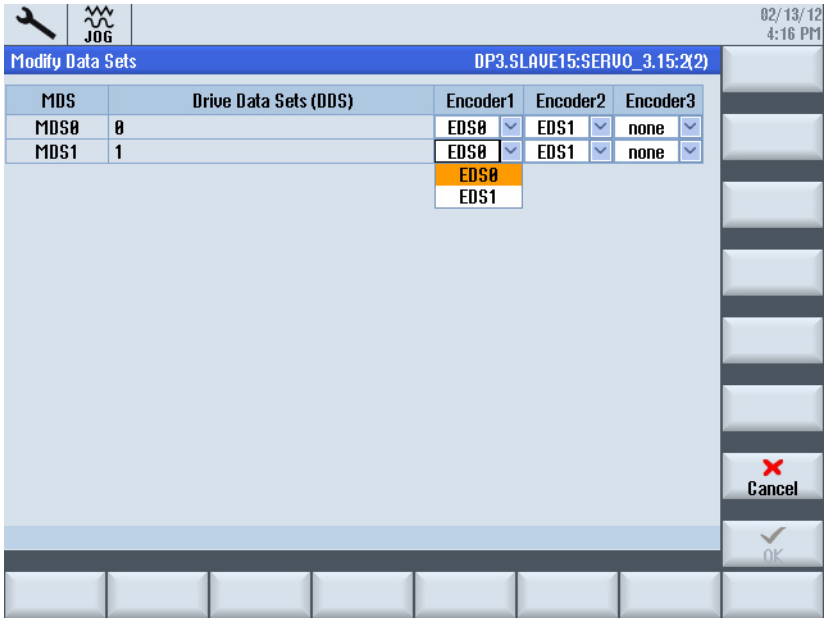
If more than one encoder is assigned to the drive object (SERVO), additional encoder data sets (EDS0 ... 2) will be created.

### Modify data set

To change the assignment of these encoder data sets for the associated encoder, select the "Modify data sets" softkey:



This activates the edit function in order to subsequently modify the assignment of EDSes to the encoders:



# Optimizing a drive

# 6

## 6.1 Control response

### Function overview

Use the "Automatic servo tuning" function to tune the current, speed and position control loop with feedforward control. The objective is the fine setting of an axis or spindle using SINUMERIK Operate:

- Performing frequency response measurements
- Calculation of the speed control loop
- Setting of current setpoint filters
- Calculation of the position control loop
- Validation of the actual behavior of the control loops
- Feedforward control type: Equivalent speed control time
- Path interpolation

### Closed-loop control

The controlled system involves three cascaded control loops:

- Current control loop
- Speed control loop
- Position control loop

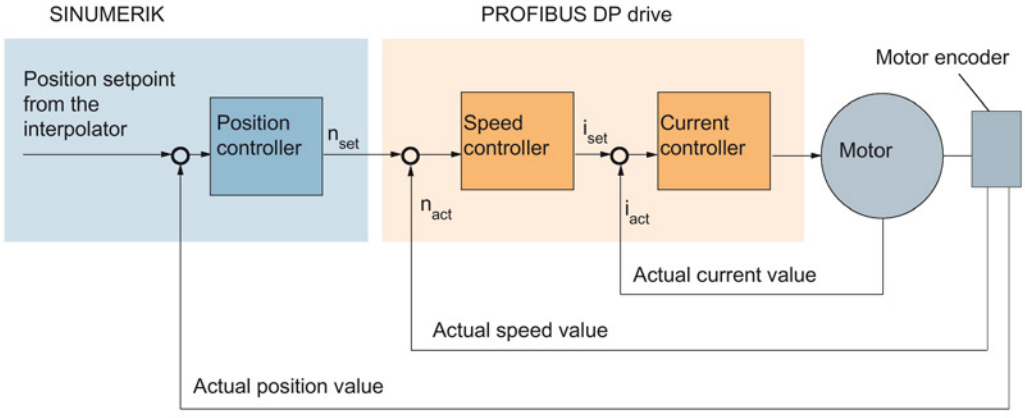



Figure 6-1 Controlled system

Each of the individual control loops is tuned successively starting with the current controller, the speed controller and finally the position controller. The dynamic machine limits for velocity, acceleration and jerk are set in the interpolator. Whereas the speed control loop is always connected via the motor encoder, the position control loop can be connected via the motor encoder or via the direct measuring system.

## Precondition

The dynamic stiffness control is set before automatic optimization.

MD32640[0] $MA_STIFFNESS_CONTROL_ENABLE = 1

This then permits a position control with a faster cycle time. When dynamic stiffness control is used, the following machine data must be set. MD32110 $MA_ENC_FEEDBACK_POL = 1

If a polarity reversal is required, the p0410 parameter "Actual encoder inversion value" must be set in the drive:

- p0410[0]=1 invert actual speed value
- p0410[1]=1 invert actual position value

If the "Automatic servo tuning" function is used for a main spindle, it is possible that no PLC signal for the servo release is present. The machine manufacturer should provide a method to activate this PLC signal for the optimization, for example, a special key combination or the setting of a flag in the PLC status.

This procedure activates specific changes of machine data. The right-hand side of the parameter value indicates how the associated change is activated:

| | |
|---|---|
| po | The changed machine data requires an "NCK Reset" to take effect. |
| re | The changed machine data must be activated by pressing the <RESET> key. |
| cf | The changed machine data must be activated by pressing the "Apply MD" softkey. |
| so | The changed machine data becomes active immediately. |

## Backing up drive data

When the drive is switched on or a "Drive Reset" performed, the SINAMICS S120 data from the saved file is loaded into the associated drive. To permanently accept changes made to the drive data after the new power on/off or after a "Drive Reset", the data must be saved.

**Note**

An "NCK Reset" does not have any effect on the drive data.

Procedure:

1. Press the "Drive parameter" softkey and then the "Save/reset" softkey.
2. Press the "Save" softkey and finally the "Drive system" softkey.

Messages in the lower left-hand area of the screen indicate the status of the save action. The following window is displayed when the save action completes: "All drive objects saved".

## References

You can find further details in the following manuals:

- SINUMERIK Operate (828D_IH9) Commissioning Manual; Chapter, "Trace"
- Basic Functions Function Manual; Velocities, Setpoint-Actual Value Systems, Closed-Loop Control (G2)
- Extended Functions Function Manual; Compensations (K3)

## See also

The following chapters describe the individual steps for optimizing control loops with SINUMERIK Operate:

- Automatic servo tuning (Page 176)
- Axis jerk (Page 189)
- Torque load (Page 197)
- Spindle tuning (Page 201)
- Circularity test (Page 211)
- Friction compensation (Page 90)

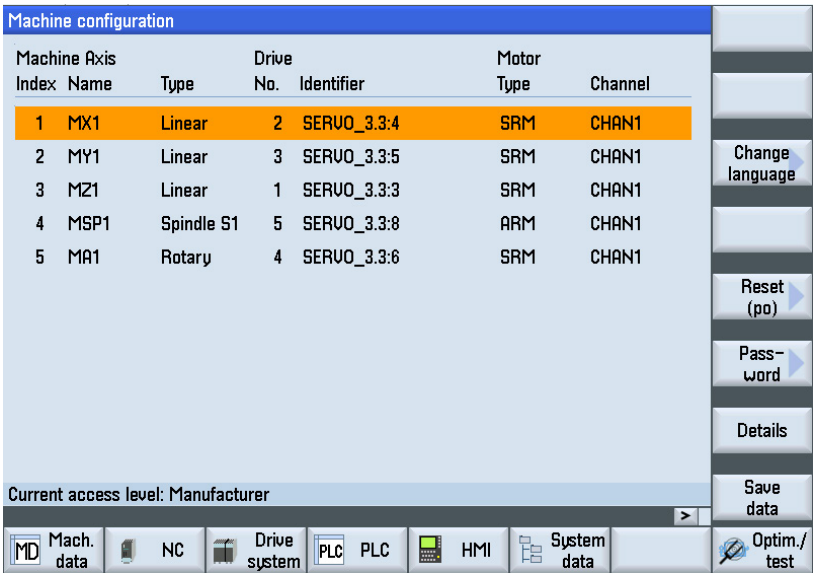
# 6.2 Automatic servo optimization

## 6.2.1 Options before the start

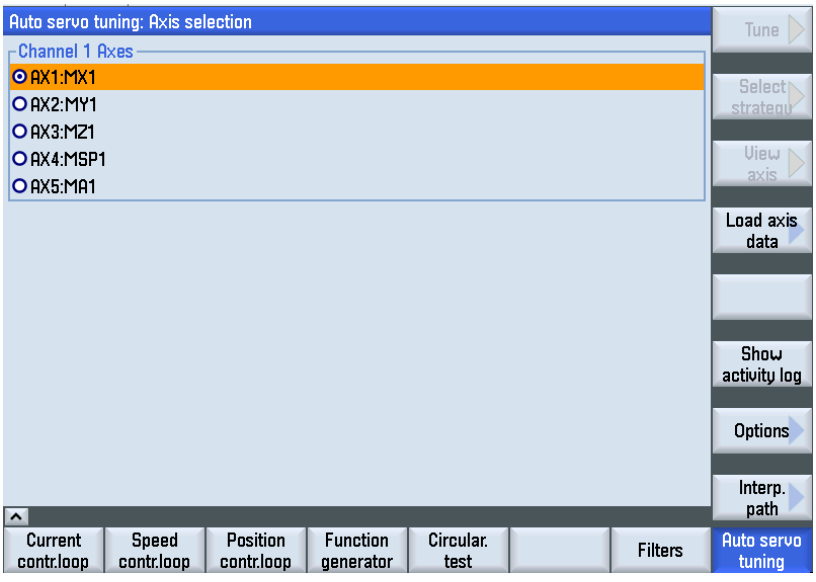
### Starting the function

Procedure:

1. Start the "Automatic servo tuning" function in the "Commissioning" operating area → "Optim./test" softkey.



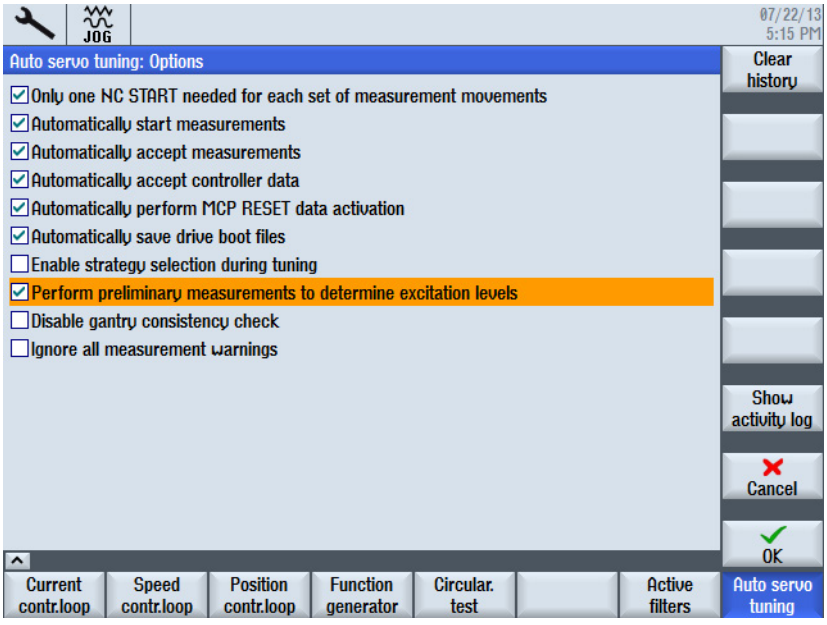

2. Use the cursor keys to mark the axis and select with the <SELECT> softkey, for example, the X1 axis:



The dialog indicates whether an axis has already been tuned. A tuned axis is marked with the date and time of the tuning.

## Selecting options

Check whether all options for an automatic procedure are set: "Options" softkey. This permits tuning by pressing a single softkey. Follow the messages and instructions shown on the screen.




The "Permit strategy selection during the tuning" option is recommended only for experts.

## 6.2.2 Selecting the tuning strategy

### Default setting for "Axis" strategy

Select a tuning strategy for axis, speed controller and position controller with the "Select strategy >" softkey. It is recommended that strategies 102, 303 and 203 are used:

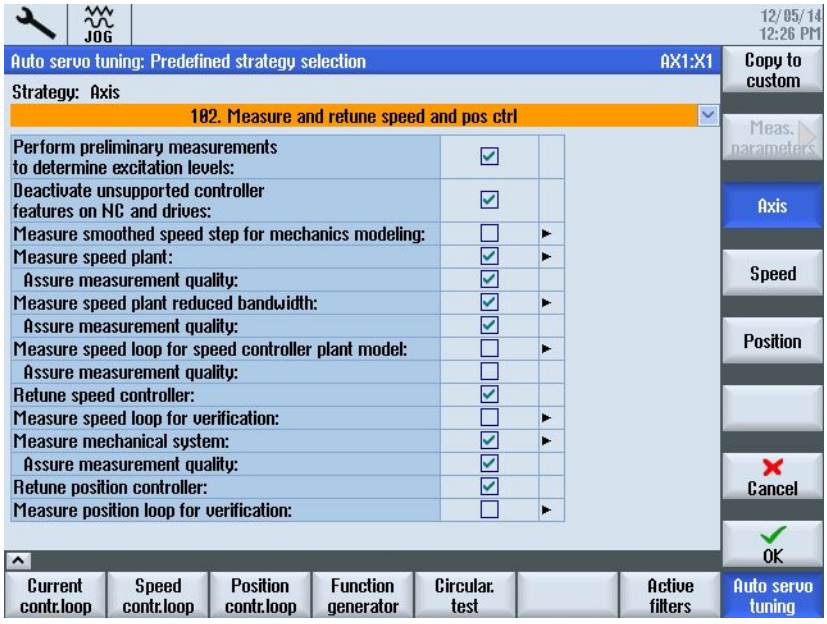



Figure 6-2 "Strategy: axis" - default setting 102

For the "Axis" strategy, select which controller should be tuned. When doing this, the tuned control loop can be measured to check the results. When selecting the "User-defined strategy (108)" for the axis, all of the options are active and can be set:

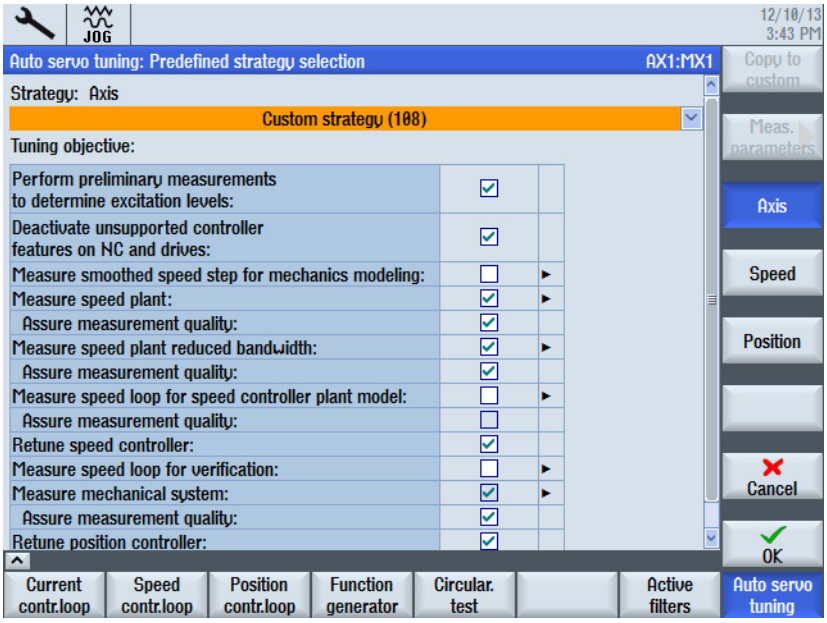



Figure 6-3 "Strategy: axis" - example (108)

Further strategies are available in the selection list and can be combined with a tuning objective. You can use the "Adapt strategy" softkey to activate and set the individual options.

## Example for "Speed": Strategy 303

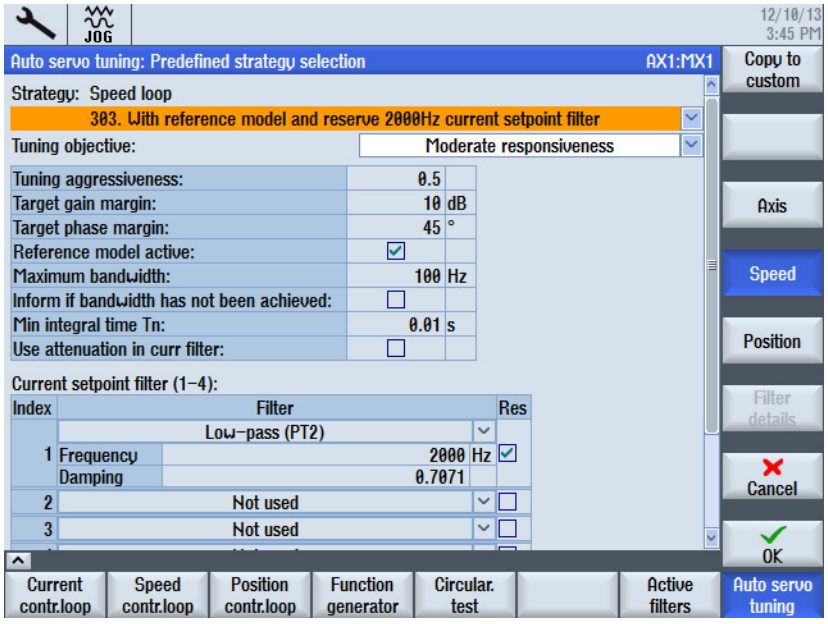



Figure 6-4 "Strategy: n/v controller" - example

The most important settings are: Tuning aggressiveness and minimum integral time Tn.

- Tuning aggressiveness:

  This parameter determines the setting of Kp and Tn based on stability limits. Depending on this setting, the values for the phase reserve and amplitude reserve are pre-assigned.

  – Default setting = 0.5
  – Min = 0 [maximum stability]
  – Max = 1 [maximum aggressiveness]

- Minimum integral time Tn:

  This parameter prevents automatic servo tuning from setting the integrator time of the speed control loop too low. If the automatic servo tuning sets a value lower than this parameter has set, the actual value used will be limited to the value set by the minimum integral time Tn.

  A higher integral time ensures a more rugged closed-loop control if changes to the weight or the moment of inertia are expected. This is recommended for direct drives and main spindles of lathes.

  – Default setting = 10 ms
  – Min = 0.5 ms
  – Max = 100 ms

## Example for "Position": Strategy 203

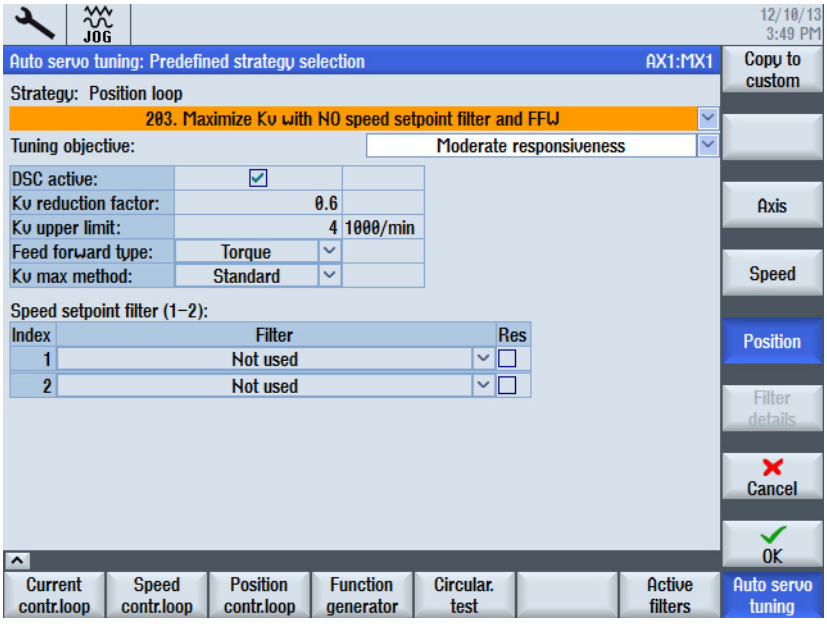



Figure 6-5 "Strategy: Position control loop" - example

The values for the Kv reduction and Kv upper limit depend on what has been selected as tuning objective.

- Kv reduction:

  This parameter specifically reduces the maximum Kv calculated by the automatic servo tuning function. No position overshoot is permitted. The maximum Kv value is a value that does not result in any overshoot. The maximum possible Kv value can be reduced to make the control more rugged against mechanical changes. A reserve of 40% is set for the tuning objective"Normal responsiveness". Further, the Kv is limited to 4 mm/min.

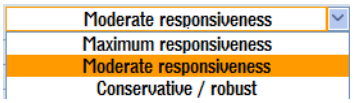
  – Default setting = 0.6
  – Min = 0.1
  – Max = 1 (no reduction)

- Feedforward control mode: Torque feedforward control is recommended.

  When telegram 136 is used, torque feedforward control allows the highest contour precision to be achieved.

## Tuning objective

The default settings for speed and position controllers are adapted by selecting the tuning objective.




- **Maximum responsiveness:**

  The speed and position controller gain (servo gain factor) is tuned with maximum values and minimum ruggedness.

  Application: High-speed machining with maximum suppression of all disturbing forces such as friction, teeth of the drive belt, strong cutting forces, when machining titanium, for example. Recommended for high-speed machining with linear motors.

  Requirement: The machine must have a rigid design; the dynamic masses do not change significantly.

- **Moderate responsiveness (= default):**

  This controller dynamic response is sufficient for the majority of machines and applications. The setting is more rugged than "Maximum responsiveness".

  Application: The inertia or the load mass of the axes does not change much and therefore this setting is suitable for numerous applications.

- **Conservative/robust responsiveness:**

  Only weak control gains are selected in order to ensure as high a level of ruggedness as possible.

  The speed controller is tuned so that it achieves maximum damping to prevent oscillations and to achieve good position controller gain.

  Application: Recommended for machines on which the mechanical axis system or the load mass can change significantly. Also suitable for axes that react with oscillations, e.g. main spindles on turning machines or large axes with high load mass.

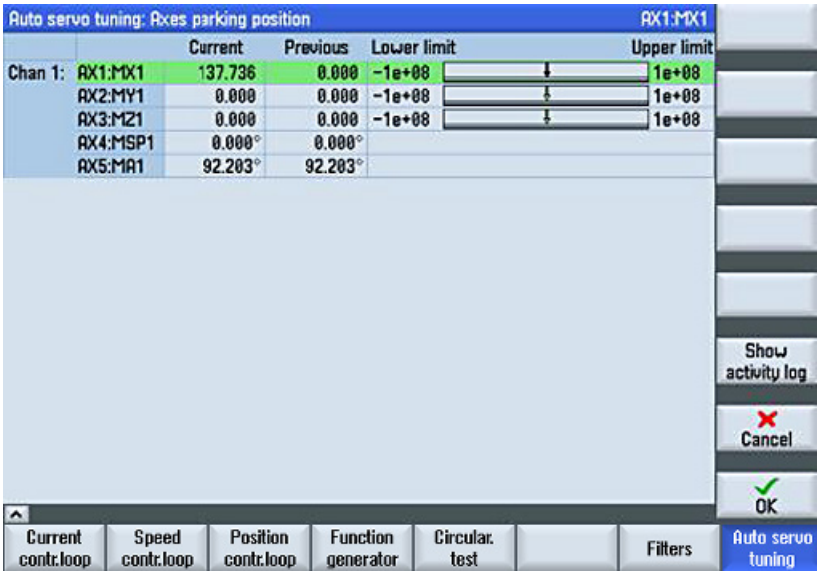
  Recommended for applications with low requirements on the machining time.

## 6.2.3 How to start the automatic servo optimization

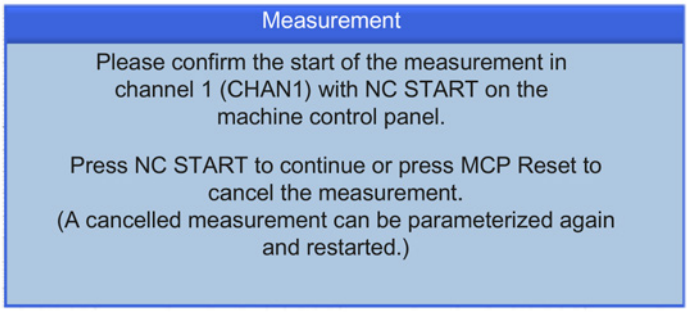
### Starting automatic servo tuning

Procedure:

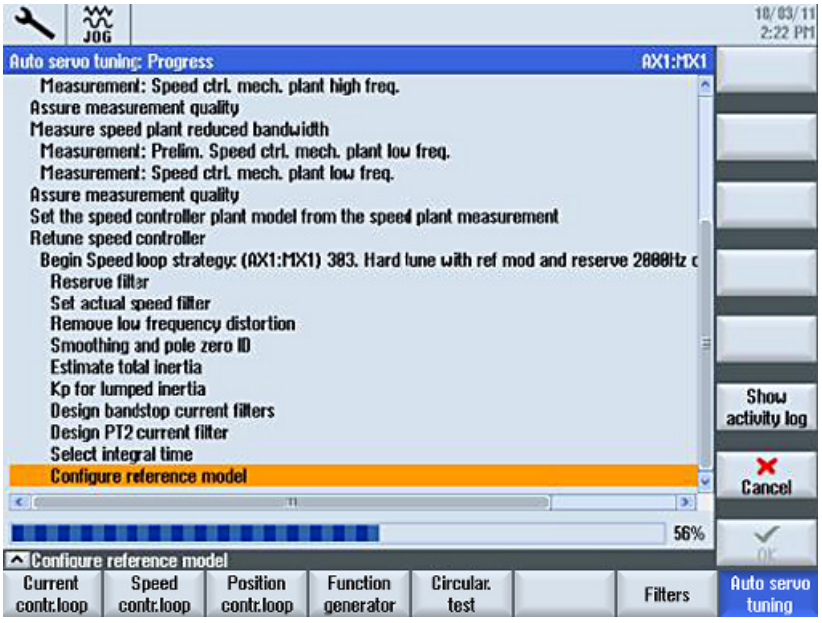
1. Ensure that the axis to be tuned is in a safe position. If this is not the case, traverse the axis in JOG mode to the center of the travel path:




2. To confirm, press "OK". The prompt to press <CYCLE START> on the machine control panel then appears:

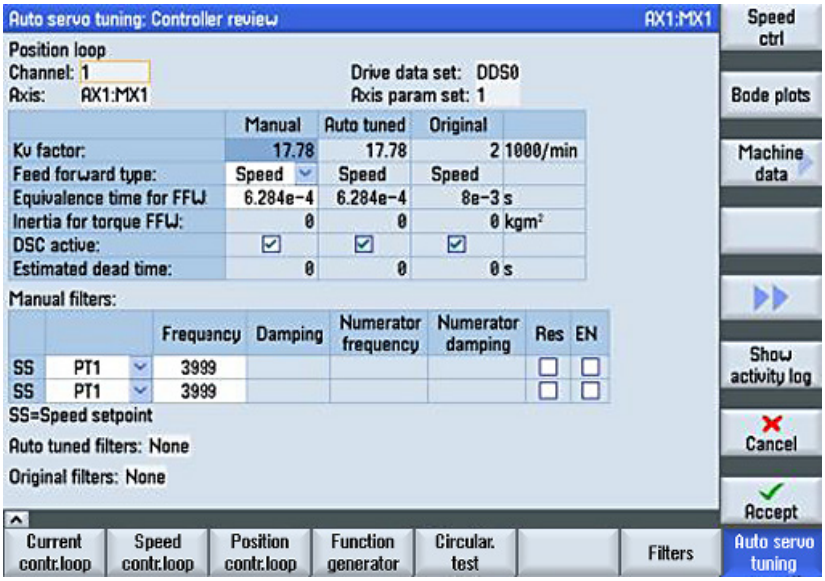


3. Further displays appear on the screen during the automatic tuning procedure. Press the <CYCLE START> key several times to start the next measurement:

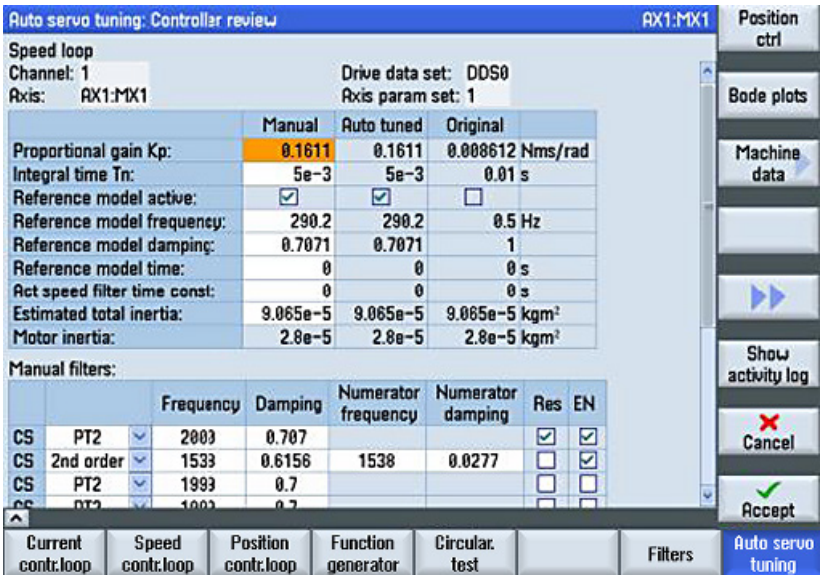



4. On completion of the tuning, the following display appears in which the new values and the original values should be checked.

   Example: Position controller

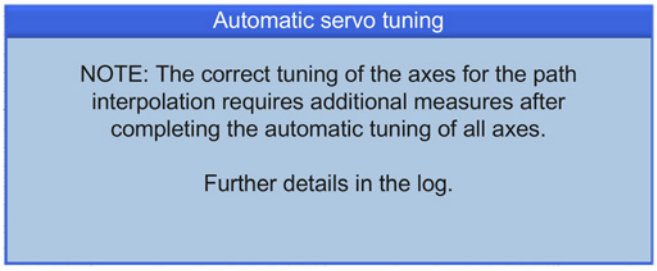


Example: Speed controller



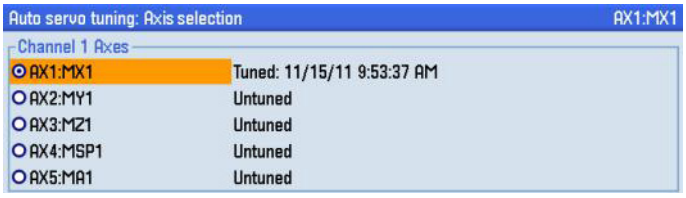

The new values are accepted when the "Accept" softkey is pressed.

5. The following message indicates that more axes must be tuned to provide a regular interpolation result:

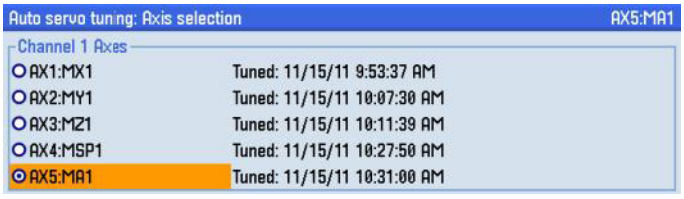



## Tuning result

The date and time of the tuning are then output:




The other axes are also tuned in the same manner:



## 6.2.4 This is how you start path interpolation

### Introduction

Interpolation means that the axes to perform the contour must exhibit the same trailing error. After the tuning of the individual axes, they have different control settings that are optimum only for the associated axis. To ensure a consistent trailing error, the control settings must be adapted so they correspond to the axis with the slowest response. This includes the servo gain factor and the feedforward control type.

### Strategy selection for the interpolation

You specify the strategy for the path interpolation here.

Default setting 1103: "Adaptation without reducing speed controller gains"

- **(1103) Path tuning without Kp reduction (recommended)**

  This strategy is strongly recommended: This is used for the standard adaptation of all feedforward control types. The axis-optimum speed controller gain Kp is not reduced.

  The following parameters are adapted:

  – Servo gain factors (MD32200)

  – Equivalent time constants for the feedforward control (MD32800 or MD32810)

  – Integral time (p1462)

  – Reference model (p1433 ff.)

  – All speed setpoint filters

  – All position setpoint filters (MD32400 and MD32900) are deactivated.

  – The feedforward control type (MD32620) is adapted to the simplest mode.

    If, for example, one axis in the interpolation group has speed feedforward control, but all the other axes have torque feedforward control, then all axes are changed to speed feedforward control. The same applies when no feedforward control is selected for one axis in the interpolation group. Then all axes are only set to position control.

- **(1104) Path tuning with user-defined Kp reduction**

  All the adaptations as in strategy 1103 are made. The user can also specify a reduction factor of the speed controller gain Kp.

  – Adapt effective Kp: ✓

  – Minimum reduction factor for Kp: Factor to which the Kp may be reduced, when required.

  – Spindle can limit other axes: If the main spindle is the weakest axis from a dynamic performance perspective, the main spindle can cause a reduction of the Kp values of the servo axes.

  – Other axis can limit the spindle: If a servo axis is the weakest axis from a dynamic performance perspective, it can also cause a Kp reduction in the spindle.

## General settings

**Settings for "Path optimal":**

- Dynamic Stiffness Control (DSC)

  MD32640 $MA_STIFFNESS_CONTROL_ENABLE is either activated or deactivated for all axes. If DSC is activated, higher $K_V$ factors are possible.

- Feedforward control type
  - Once all axes with torque feedforward control have been tuned, this setting is retained for the path interpolation.
  - If the feedforward control for the axes is set differently to 3 or 4, MD32620 $MA_FFW_MODE = 3 will then be set for all axes.
  - If one axis without feedforward control has been tuned, MD32620 $MA_FFW_MODE = 0 will be set for all axes.

**Settings for "Axis optimal":**

- The $K_V$ gain factor is calculated; the lowest setting is used for all axes: the $K_V$ factor is always adapted!
- Kp proportional gain / moment of inertia for the later adaptation of the axes.
- The Tn delay time is adapted for all axes.
- The reference model (speed controller) is activated, i.e. it is adapted.
- Equivalent time constant for the speed control loop feedforward control is calculated; the highest setting is used for all axes.

MD32402 $MA_AX_JERK_MODE = 0 Filter type for axial jerk limitation is not activated.

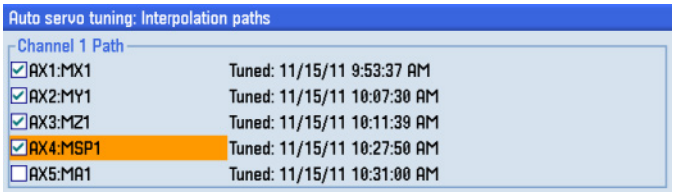
## Path interpolation

In the next step, the axes for the path interpolation are selected.

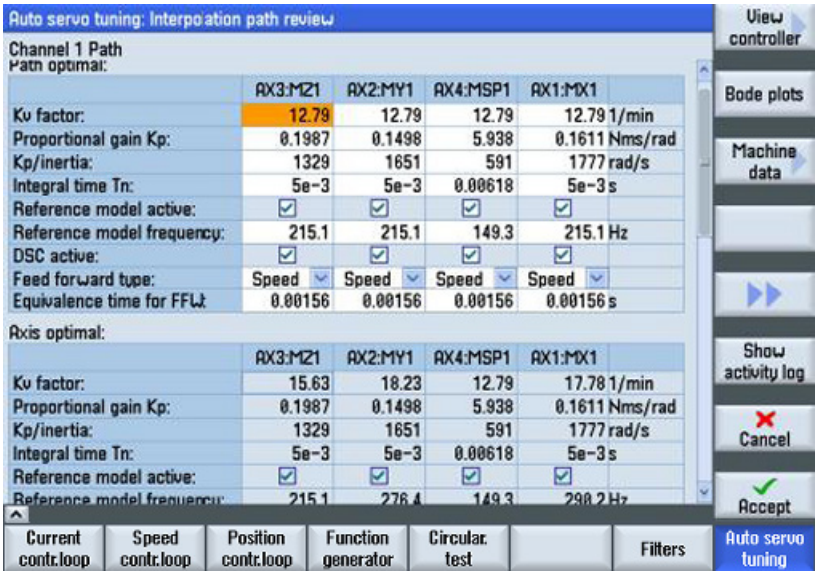
Procedure:

1. Press the "Path interpolation" softkey. The axes that interpolate together are assigned to "Channel 1".
2. Press the "Edit & tune" softkey.
3. Select only those axes that can interpolate together.

   X, Y, Z and the spindle are selected for the following example:



4. The new setting is calculated and displayed for validation:




5. Select "Accept" and confirm with OK.




This completes the automatic tuning.

# 6.3 Optimizing the axis jerk

## 6.3.1 Checking the axis jerk

### Jerk limitation

The jerk-limited acceleration is optimized in the next step: The jerk is used to smooth the acceleration and response behavior by applying a time for changing the acceleration.

The jerk is defined in units/s$^3$, i.e. acceleration [units/s$^2$] / time [s].

The time during which the acceleration change occurs is defined as: t [s] = acceleration [units/s$^2$] / jerk [units/s$^3$].

Axes do not need to have the same jerk value. The result of a jerk limited acceleration is rounded corners at the acceleration-phase start and end of the velocity profile and smoothed movements.

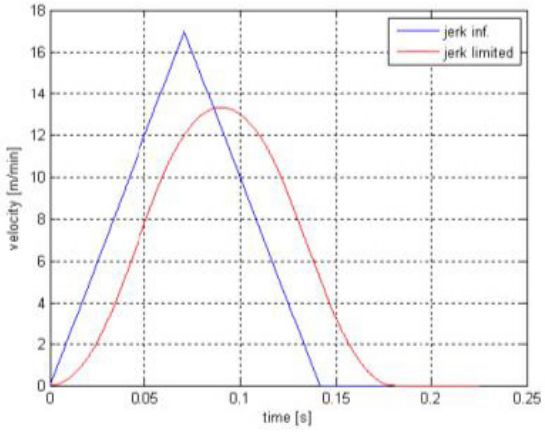
- Jerk limitation is active when "SOFT" is programmed.
- Jerk limitation is deactive when "BRISK" is programmed.

### Machine data for jerk

It is recommended that the MD20150[20] = 2 default setting, i.e. "SOFT", is used.

- MD20150[20] = 1 then "BRISK" is active in the switch-on state (acceleration without jerk limitation).
- MD20150[20] = 2 then "SOFT" is active the switch-on state (acceleration with jerk limitation).

Example:




blue Acceleration without jerk limitation

red Acceleration with jerk limitation

### Positioning behavior of the axes

The Trace function over position setpoint steps using a part program with different feeds can be used to check the positioning behavior.

- If a direct scale is used, the response behavior of the carriage can be seen easily.
- If a motor encoder is used to close the position control loop (indirect feedback), the response behavior of the carriage for the closed position control loop with the Trace function cannot be determined.

In general, jerk values between 20 and 100 are used for machines with SINUMERIK 828D. The test program should move the axis over an adequate distance so that the programmed feedrate velocity can be attained.

A delay time, for example 0.5 seconds, after the feedrate velocity and the position setpoint is attained, should be used. The first programmed feedrate velocity should be 50% of the maximum axis feedrate. This behavior should be checked for all increments of the feedrate override up to the maximum feedrate velocity.

## 6.3.2 Parts program for axis jerk

### Parts program for checking

The following NC program can be used in automatic or MDA operation depending on the configuration of the deployed machine. The program contains commands for the activation/deactivation of the trace trigger:

```
FFWON

SOFT

$AN_SLTRACE=0; Reset Start Servo-Trace-Trigger

LAB:

G01 Y210 F10000

G04 F0.5

$AN_SLTRACE=1; Start Servo-Trace-Trigger

Y260

G04 F0.5

GOTOB LAB

M30
```

View the trace results:

Select <MENU SELECT> → "Diagnosis" operating area, menu forward key → "Trace" softkey and press <CYCLE START>:

The axis moves and the selected signals are monitored for five seconds. The result is then displayed.

**Machine data:**

To check the positioning behavior without the effect of the position controller, the $K_v$ position-control gain factor should be set to 0. Specify positioning tolerances to prevent errors during the test:

- MD32200 $MA_POSCTRL_GAIN = 0
- MD36012 $MA_STOP_LIMIT_FACTOR = 100
- MD36400 $MA_CONTOUR_TOL = 20

### Initializing the trace

The "Start trace" softkey is used to start and initialize the tracing. This depends on the selection defined in the "Settings" dialog. In this example, the trace is started with a parts-program command. Press the "Start trace" softkey to initialize the trace.

The trace is now initialized and will be started when the trigger is set by the part program.

## 6.3.3 Selecting trace signals

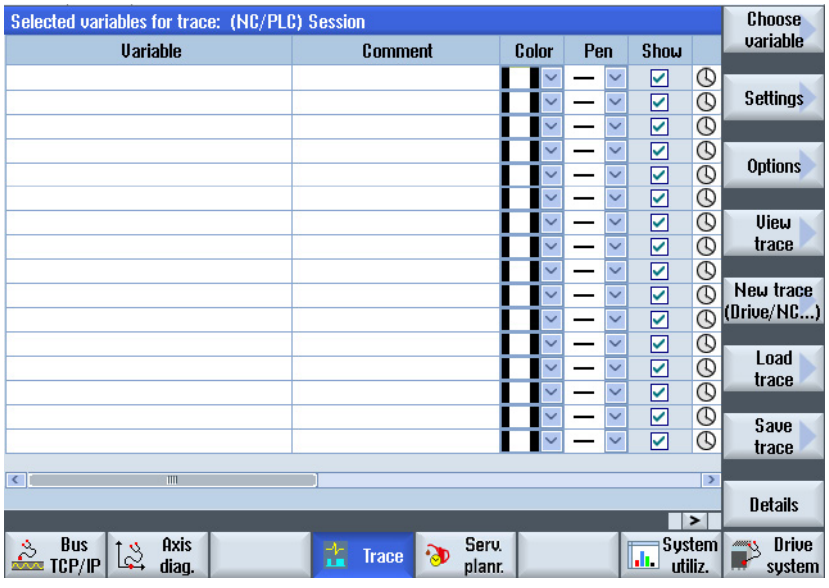
### Selecting trace signals

A trace of the position setpoints and actual position values is required.

Procedure:

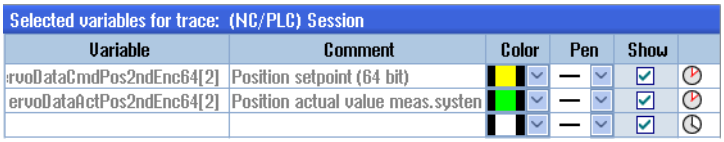
1. Use the following key combination to invoke the Trace function:

   <MENU SELECT> → "Diagnostics" operating area → Menu forward key → "Trace" softkey




2. The required variables are selected: "Select variable" softkey.

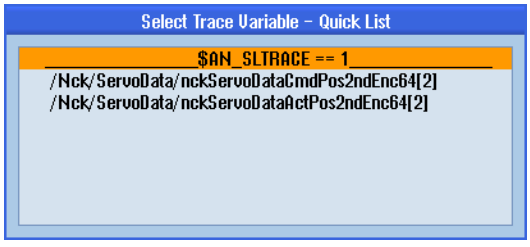
3. Press the "Filter/search" softkey to limit the list of variables. The following filters are available:
   - System variables
   - NC
   - Axis
   - Channel
   - Mode groups
   - Servo
   - Global user data (GUD)
   - PLC
   - PLC alarms
   - All (no filter)
4. Set the filter to "Servo" and select the position setpoint and actual position-value variables, e.g. for the Y axis:



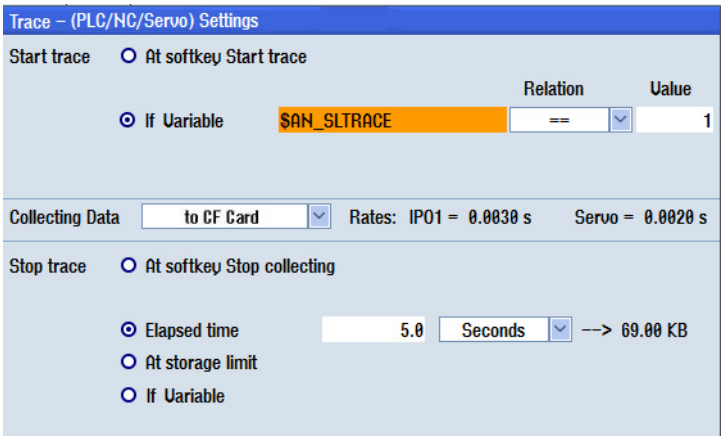

## Trace duration and start conditions

The trace can be started automatically or manually. The trace duration can also be set. In the following example, the trace is started with a parts-program command. The "Quick list" softkey can be used to select the variable.

Select the $AN_SLTRACE==1 option to start the trace with a variable:

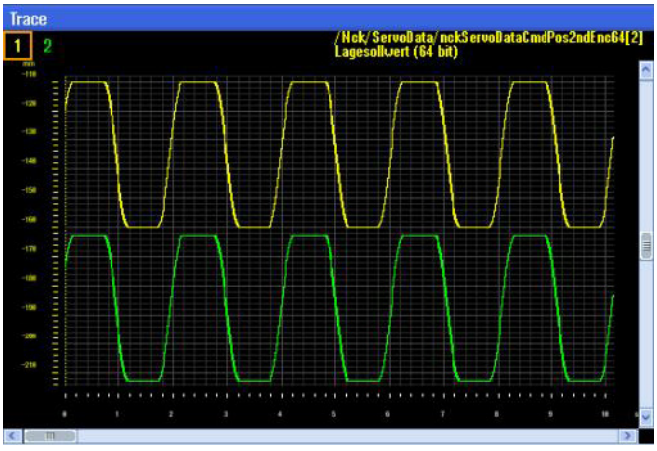


It is also possible to initiate the trace from a value of the previously selected signals or with a system variable made available with the "Add variable" softkey:




## Trace result

The axis moves and the selected signals are monitored for five seconds. The result is then displayed:

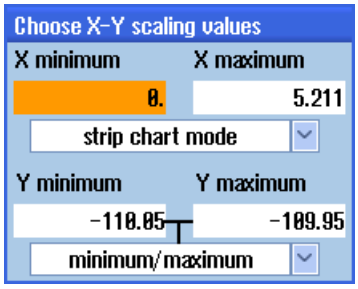


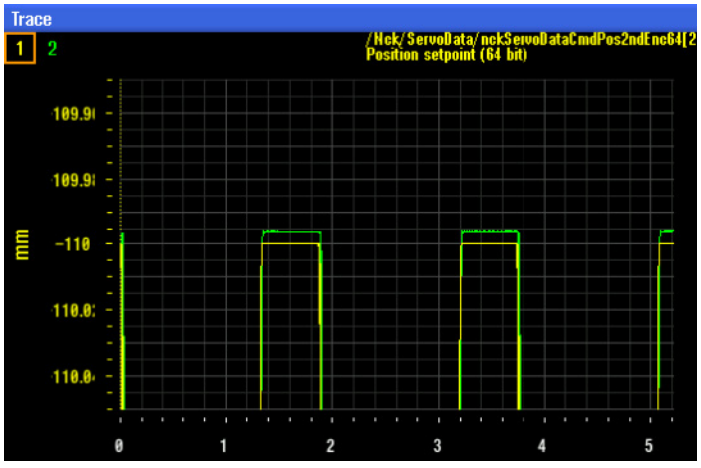

## Setting the trace scale

It is necessary to view that part of the trace in which the axis reaches the position setpoint. To view this in detail, set the trace scale: "Display trace" softkey → "Scale" softkey

The following example shows the setting of the scale to view the axis positioning at -110 mm. The entered values produce a 100-micrometer window around the target position of -110 mm. Depending on the used machine/axis, a different scale may be necessary.

If necessary, the X minimum/maximum values can also be adapted:




View in detail:



Position setpoint (yellow) and actual position (green) do not match because the position controller is not active.

## 6.3.4 Optimizing the jerk setting

### Optimization objective

The jerk must be set so that the actual position is not exceeded. This is done by setting a jerk value that is too high.

A jerk value that is too low can also be disadvantageous for the machine because it affects the time required for positioning an axis. Without the position controller, the axis is positioned without overshooting.

The remaining positioning error results from the position controller being deactivated and the axis moves only with feedrate signals. An overshooting on reaching the position that results from the mechanical system can be prevented by using a jerk limitation.

The following machine data is first checked and set:

- MD32200 $MA_POSCTRL_GAIN = 0
- MD32610 $MA_VELO_FFW_WEIGHT = 1
- MD32620 $MA_FFW_MODE = 3
- MD32810 $MA_EQUIV_SPEEDCTRL_TIME = as optimized

The following examples show the effects of different jerk values on the positioning of an axis.

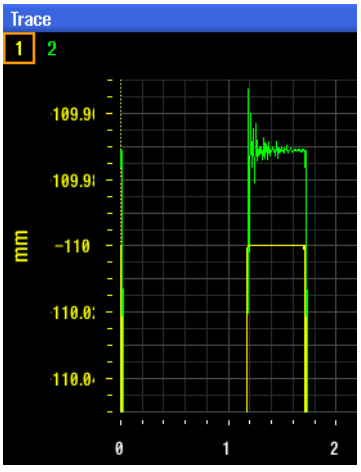
## Examples

The MD32431 axis machine data is used to set the jerk.

**Example 1:** MD32431 $MA_MAX_AX_JERK = 600 ⇒ jerk too high

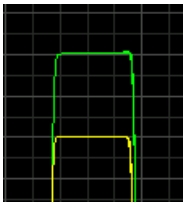
1. Restart the trace and then execute the part program.
2. Press the "Start trace" softkey and then the <RESET> and <CYCLE START> keys.
3. Correct the overshoot by increasing MAX_AX_JERK.

This diagram shows the jerk increase until an overshoot caused by the mechanical system.



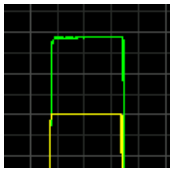

**Example 2:** MD32431 $MA_MAX_AX_JERK = 2 ⇒ jerk too low

A jerk that is too low causes the positioning contour to be rounded.



**Example 3:** MD32431 $MA_MAX_AX_JERK = 65 ⇒ jerk optimized

If the jerk is optimized, there is no overshoot or it is small and the positioning contour is sharp.



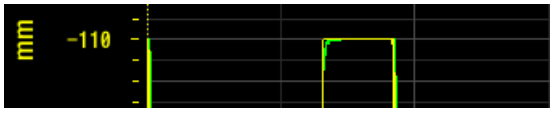
## Example: Positioning with precontrol and active position controller

If the jerk has been optimized, the position controller and the precontrol must be activated.

Set the machine data as required:

- MD20150[23] =2 The precontrol can then be released as default setting.
- MD32200 $MA_POSCTRL_GAIN = as optimized
- MD32610 $MA_VELO_FFW_WEIGHT = 1.0
- MD32620 $MA_FFW_MODE = 3
- MD32810 $MA_EQUIV_SPEEDCTRL_TIME = as optimized
- MD32431 $MA_MAX_AX_JERK = 65
- MD36012 $MA_STOP_LIMIT_FACTOR = original value
- MD36400 $MA_CONTOUR_TOL = original value

A good positioning without overshoot is achieved when the precontrol and jerk have been optimized correctly:



# 6.4 Torque load

## Checking the maximum axis acceleration

To ensure that the torque limit is not reached, after optimizing the Kv factor and the feed precontrol, check the torque loading under unfavorable conditions:

- At high velocity
- For high loading

The "Trace" function is used to measure the velocity setpoint of the active measuring systems and the percentage torque load with the "Load (m_set / m_set_limit)" trace signal. If the "Load (m_set / m_set_limit)" trace signal is set to 100%, the drive operates with current limitation. Change the measurement parameters in the trace to display the torque loading.

MD32300 $MA_MAX_AX_ACCEL defines the maximum axis acceleration. The default setting is [1 m/s$^2$ (metric), 39.37 in/s$^2$ (inch) and 2.77 rev/s$^2$ (rotary)]. Each axis can have a different setting for the acceleration.

If permitted by the machine manufacturer, MD32300 can be set so that the torque loading during the acceleration phase in the worst case lies between 80% and 90% of the limit value, depending on the size of the desired safety factor.

**Note**

The machine mechanical system limits the maximum axis acceleration. The machine manufacturer must define the required setting value! If this value is not defined, use the default setting unless the machine manufacturer permits an increased value.

## Example: Test program

In this test program, the axes should be moved within an adequate distance so that the programmed rapid-traverse velocity can be attained, e.g. 300 mm. To do this, change the test program as follows:

```
SOFT

$AN_SLTRACE=0; Reset Start Servo-Trace-Trigger

LAB:

G0 X10

$AN_SLTRACE=1; Start Servo-Trace-Trigger

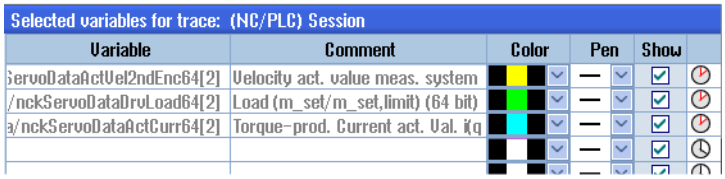
X310

GOTOB LAB

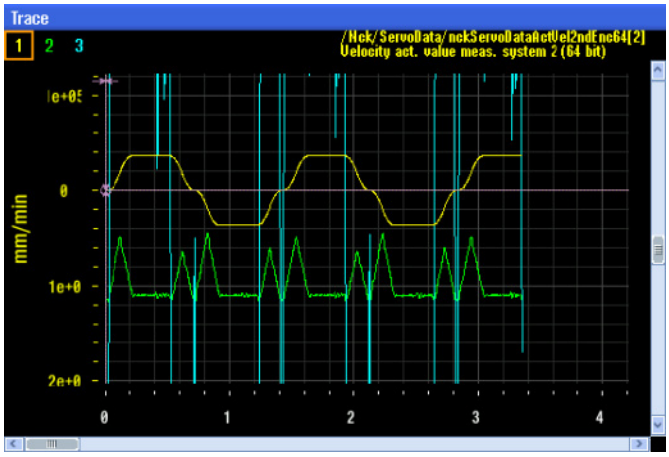
M30
```

Select one of the following signals for monitoring:

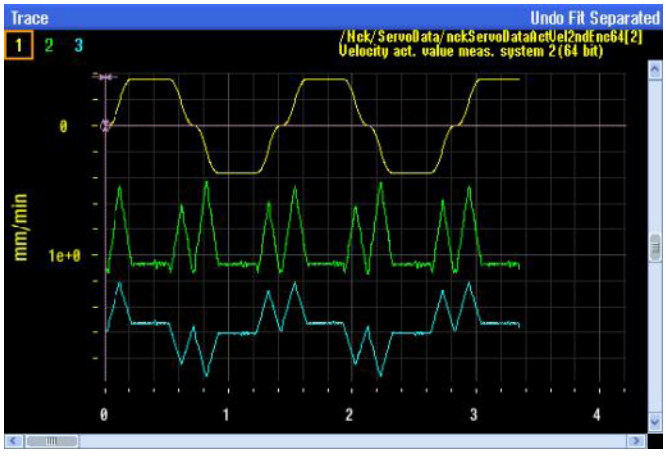
- Actual velocity measuring system 2
- Load (m_set/m_set_limit)
- Torque-generating actual current value i(q)




"Start trace" softkey → <RESET> → <CYCLE START>:




The three traces are displayed together; press the "Adapt all" softkey:

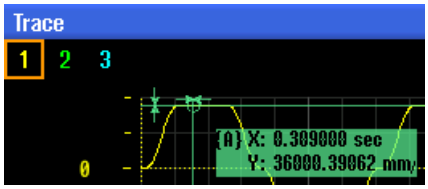



To recognize the maximum values of the three traces, press the "Cursors" softkey and position it on the associated trace to mark the maximum values: "Cursors" softkey → "Cursor A" softkey.

To check the maximum values of the three traces, press the "Cursors" softkey and position the cursor on the associated trace to read the maximum values of the following quantities:

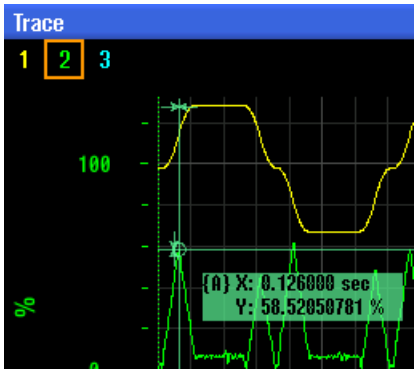
- **Velocity**

  The cursor positioned over the trace shows that the attained maximum velocity is 36 m/min:



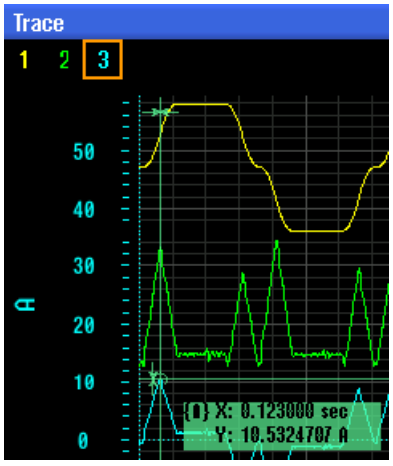

- **Load**

  The cursor positioned over the trace shows that the attained maximum load is 58.52 %/min:




- **Current**

  The cursor positioned over the trace shows that the attained maximum current is 10.53 A:



## Summary

The maximum velocity of 36 m/min is attained with a maximum load of 58.52%. The "Load (m_set/m_set_limit)" signal selection shows the percentage load of the active current limit. The active current limit is: p0640 = [A effective value]

p0640 is limited to p0338 motor current limit / Mot I Max [A effective value] or r0209 drive current limit / PU I Max [A effective value] depending on which value is lower.

In this example, p0640 = 18 A

Consequently the load is 18 A x 58,52/100 = 10,53 A

The test was performed on the Y axis of a milling machine with a dividing unit on the table. The load of 58% permits an additional weight to be added without problem. The load should not exceed 85%. To affect the load, the acceleration can be changed.

# 6.5 Optimize the spindle

## 6.5.1 Set the machine data for the spindle

### Checking the machine data settings

Match the machine data default settings as well as all relevant parameters for spindle/axis interpolation with the machine data in the following table. The example is true for a spindle motor without mechanical gearbox stages with 8000 rpm.

The machine data can be initialized with the suggested values shown in the table. The values vary depending on the application.

| Number | Designation | Value | Meaning |
|---|---|---|---|
| MD30300 | $MA_IS_ROT_AX | 1 | Rotary axis / spindle |
| MD30310 | $MA_ROT_IS_MODULO | 1 | Modulo conversion for rotary axis/spindle |
| MD30320 | $MA_DISPLAY_IS_MODULO | 1 | 360-degree modulo display for rotary axis or spindle |
| MD32000 | $MA_MAX_AX_VELO | 8000 | Maximum axis velocity |
| MD32010 | $MA_JOG_VELO_RAPID | 60 | Rapid traverse in jog mode |
| MD32020 | $MA_JOG_VELO | 10 | Jog axis velocity |
| MD32040 | $MA_JOG_REV_VELO_RAPID | 60 | Revolutional feedrate in JOG mode with rapid traverse override |
| MD32050 | $MA_JOG_REV_VELO | 10 | Revolutional feedrate for jog |
| MD32200[0] | $MA_POSCTRL_GAIN | x | $K_V$ factor: as optimized |
| MD32200[1] | $MA_POSCTRL_GAIN | x | |
| MD32620 | $MA_FFW_MODE | 3 | Feedforward control type |
| MD32640 | $MA_STIFFNESS_CONTROL_ENABLE | 1 | Dynamic stiffness control |
| MD32810[0] | $MA_EQUIV_SPEEDCTRL_TIME | x | Equivalent time constant speed control loop for feedforward control: as optimized |
| MD32810[1] | $MA_EQUIV_SPEEDCTRL_TIME | x | |
| MD33000 | $MA_FIPO_TYPE | 3 | Fine interpolator type |
| MD34000 | $MA_REFP_CAM_IS_ACTIVE | 0 | Axis with reference point cams |
| MD34020 | $MA_REFP_VELO_SEARCH_CAM | 30 | Reference point approach velocity |
| MD34020 | $MA_REFP_VELO_SEARCH_CAM | 30 | |
| MD34040[0] | $MA_REFP_VELO_SEARCH_MARKER | 30 | Creep velocity |
| MD34040[1] | $MA_REFP_VELO_SEARCH_MARKER | 30 | |
| MD34060[0] | $MA_REFP_MAX_MARKER_DIST | 370 | Maximal distance to the reference mark |
| MD34060[1] | $MA_REFP_MAX_MARKER_DIST | 370 | |
| MD35000 | $MA_SPIND_ASSIGN_TO_MACHAX | 1 | Assignment of spindle to the machine axis |
| MD35100 | $MA_SPIND_VELO_LIMIT | 8000 | Maximum spindle speed |
| MD35110[0] | $MA_GEAR_STEP_MAX_VELO | 8000 | Maximum speed for gear stage change |
| MD35110[1] | $MA_GEAR_STEP_MAX_VELO | 8000 | |
| MD35130[0] | $MA_GEAR_STEP_MAX_VELO_LIMIT | 8000 | Maximum speed of gear stage |

| Number | Designation | Value | Meaning |
|---|---|---|---|
| MD35130[1] | $MA_GEAR_STEP_MAX_VELO_LIMIT | 8000 | |
| MD35200[0] | $MA_GEAR_STEP_SPEEDCTRL_ACCEL | x | Acceleration in speed control mode: as optimized |
| MD35200[1] | $MA_GEAR_STEP_SPEEDCTRL_ACCEL | x | |
| MD35210[0] | $MA_GEAR_STEP_POSCTRL_ACCEL | x | Acceleration in position control mode: as optimized |
| MD35210[1] | $MA_GEAR_STEP_POSCTRL_ACCEL | x | |
| MD35500 | $MA_SPIND_ON_SPEED_AT_IPO_START | 2 | Feed enable with spindle in setpoint range |
| MD35550[0] | $MA_DRILL_VELO_LIMIT | 4000 | Maximum speeds for tapping |
| MD35550[1] | $MA_DRILL_VELO_LIMIT | 4000 | |
| MD36000 | $MA_STOP_LIMIT_COARSE | 0,4 | Coarse exact stop |
| MD36010 | $MA_STOP_LIMIT_FINE | 0,1 | Fine exact stop |
| MD36030 | $MA_STANDSTILL_POS_TOL | 5 | Standstill tolerance |
| MD36040 | $MA_STANDSTILL_DELAY_TIME | 1 | Zero speed monitoring delay time |
| MD36050 | $MA_STOP_ON_CLAMPING | 1 | Clamping tolerance |
| MD36060 | $MA_STANDSTILL_VELO_TOL | 2 | Threshold velocity / speed "Axis/spindle stationary" |
| MD36200 | $MA_AX_VELO_LIMIT | 8800 | Threshold value for velocity monitoring |
| MD36200 | $MA_AX_VELO_LIMIT | 8800 | |
| MD36300[0] | $MA_ENC_FREQ_LIMIT | 1000000 | Encoder limit frequency |
| MD36300[1] | $MA_ENC_FREQ_LIMIT | 1000000 | |
| MD36400 | $MA_CONTOUR_TOL | 30 | Contour monitoring tolerance band |
| p1433 | SPEED_CONTROLLER_REFERENCE_MODEL_NATURAL_FREQUENCY | x | Speed controller reference model natural frequency / n_reg RefMod fn: as optimized |

## 6.5.2 Spindle: Checking the speed controller

### Checking the acceleration of the speed controller

The speed controller acceleration is checked. First, the acceleration time must be specified without effects by the NC.

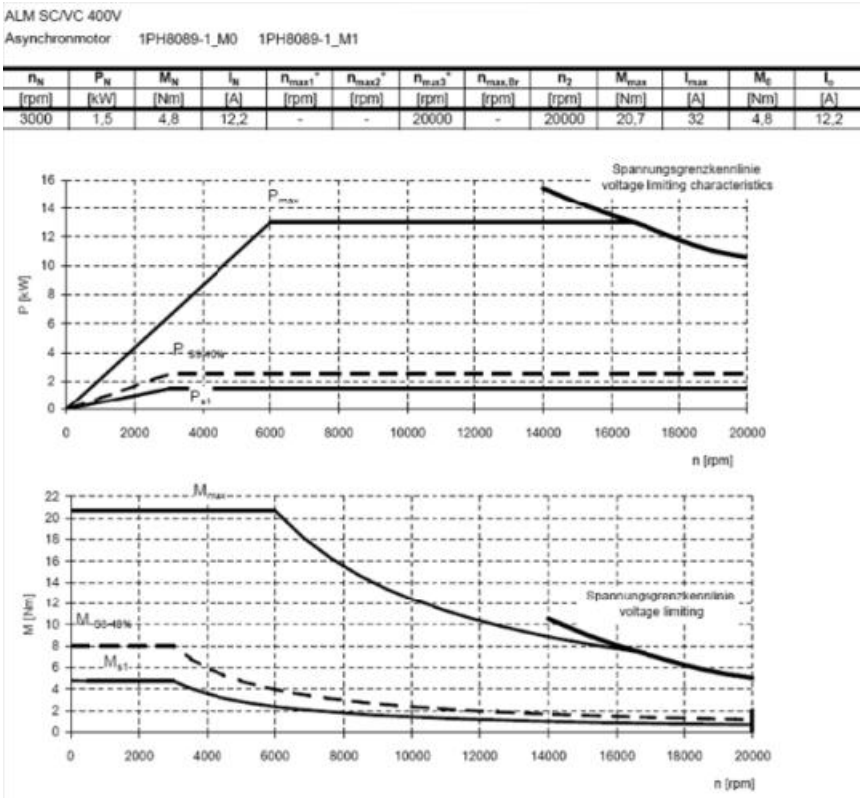
Set the following machine data:

- MD35200[0] $MA_GEAR_STEP_SPEEDCTRL_ACCEL = 9999
- MD35200[1] $MA_GEAR_STEP_SPEEDCTRL_ACCEL = 9999

  ( t=V/a t=9000/60s/9999 t=15 ms, consequently no effect).

## Motor data sheet

Example: 1PH8089-1VM02-0MG1 induction motor




| $n_N$ | $P_N$ | $M_N$ | $I_N$ | $n_{max1}$* | $n_{max2}$* | $n_{max3}$* | $n_{max,0r}$ | $n_2$ | $M_{max}$ | $I_{max}$ | $M_0$ | $I_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| [rpm] | [kW] | [Nm] | [A] | [rpm] | [rpm] | [rpm] | [rpm] | [rpm] | [Nm] | [A] | [Nm] | [A] |
| 3000 | 1,5 | 4,8 | 12,2 | - | - | 20000 | - | 20000 | 20,7 | 32 | 4,8 | 12,2 |

Figure 6-6 1PH8 induction motor

Accelerating and decelerating with the original data of the motor corresponds to the S1 curve.

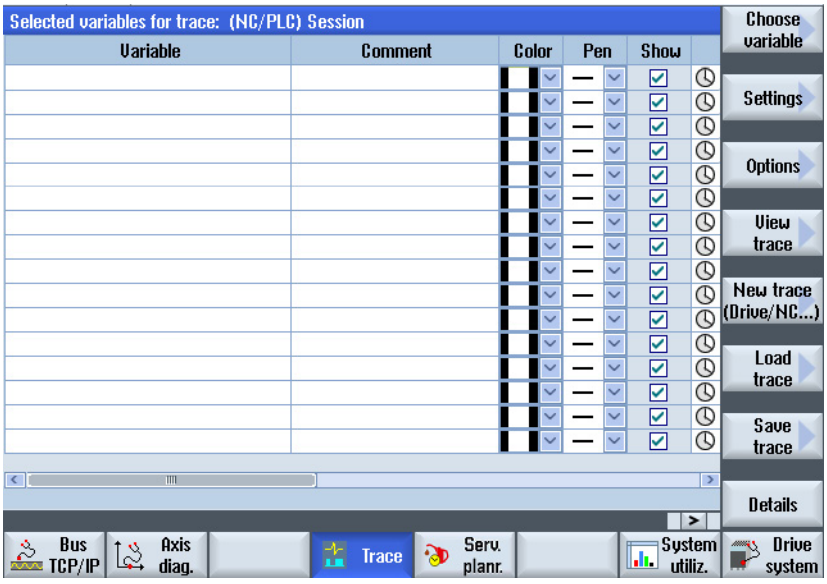
| Number | Designation | Value |
|---|---|---|
| p0640 | Current limit (p0305 motor rated current * 1.5) | 17.7 A |
| p1520 | CO: Torque limit upper/motorized M_max upper/mot | 4.8 Nm |
| p1521 | CO: Torque limit lower/regenerative / M_max lower/gen | -4.8 Nm |
| p1530 | Power limit motorized / P_max mot | 1.5 kW |
| p1531 | Power limit regenerative / P_max gen | -1.5 kW |

## Trace recording

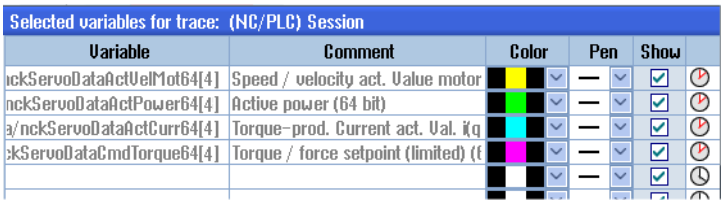
Procedure:

1. Use the following key combination to invoke the Trace function:

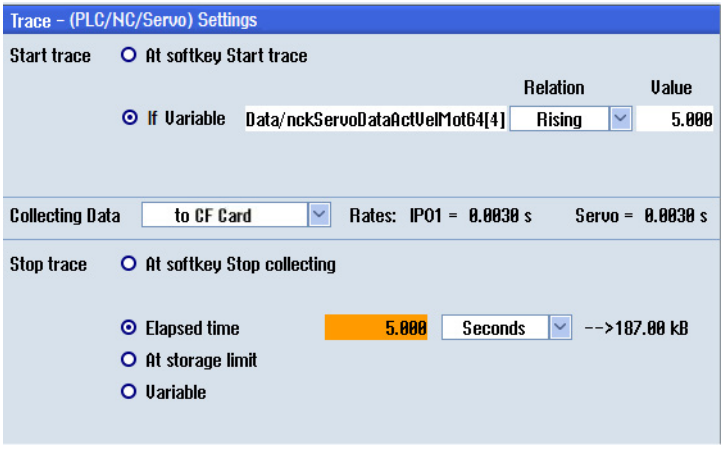
   <MENU SELECT> → "Diagnostics" operating area, menu forward key → "Trace" softkey




2. To select the required variables: "Select variable" softkey
   - Actual motor speed/velocity
   - Active power
   - Torque-generating actual current value i(q)
   - Torque/force setpoint (limited)
3. The required variables are selected: "Select variable" softkey.
4. Press the "Filter/search" softkey to limit the list of variables.
5. Ensure that the spindle is selected when you add the variables:



6. Press the "Settings" softkey.




7. Confirm with "OK".
8. Enter the following program in the MDA:

```
S8000 M03

G04 F5

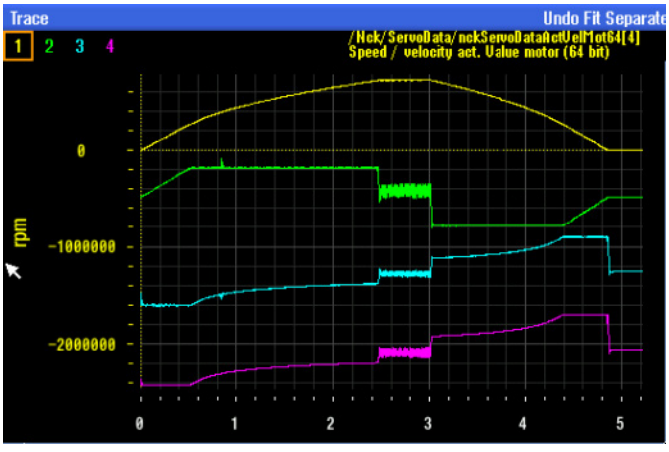
M30
```

9. Initialize the trace and execute the program:

   "Trace" softkey → "Display trace" softkey → "Start trace" softkey → <RESET> key → <CYCLE START> key

## Evaluation

When the trace function completes, press the "Adapt all" softkey:

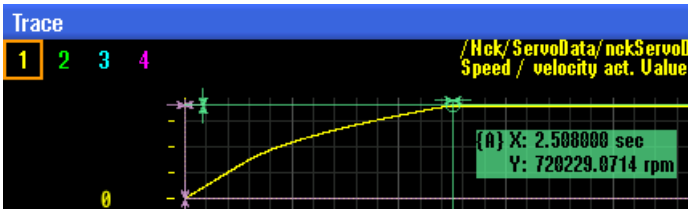


Use the cursor keys to set the acceleration time.

- **Trace selection: Actual speed (Trace 1)**

  The cursor is positioned at the trace start and at the point where the speed setpoint is attained.

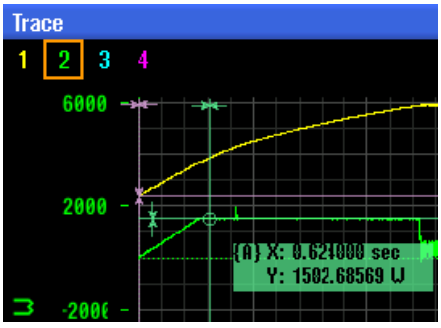
  Time needed for acceleration to the speed setpoint = 2.5 seconds




- **Trace selection: Actual speed (Trace 2)**

  The power used to accelerate the motor can be seen on Trace 2:

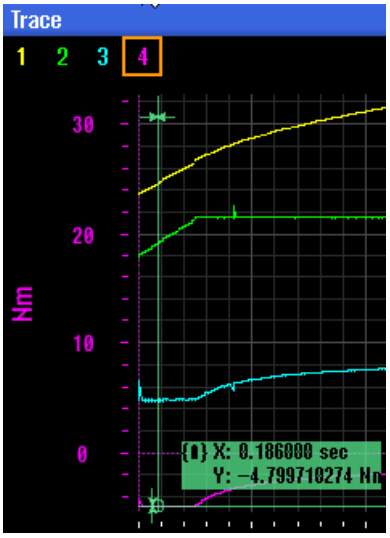
  Maximum available power = 1.5 kW = S1 curve




- **Trace selection: Torque setpoint (Trace 4)**

  The torque setpoint can be seen on Trace 4:

  Torque setpoint = 4.8 Nm = S1 curve



## Checking the acceleration

The acceleration/deceleration can be changed using the motor datasheet through to the maximum values, although this depends on the associated application. Enter appropriate values for the application. In our example, a short acceleration/braking time is requested.

| Number | Designation | Value |
|---|---|---|
| p0640 | Current limit (p0305 motor rated current * 1.5) | 32 A |
| p1520 | CO: Torque limit upper/motorized M_max upper/mot | 20.7 Nm |
| p1521 | CO: Torque limit lower/regenerative / M_max lower/gen | -20.7 Nm |
| p1530 | Power limit motorized / P_max mot | 13 kW |
| p1531 | Power limit regenerative / P_max gen | -13 kW |

Enter the maximum values from the motor datasheet in the appropriate drive data: e.g. 1PH8089-1_M0, 1PH8089-1_M1 asynchronous motor

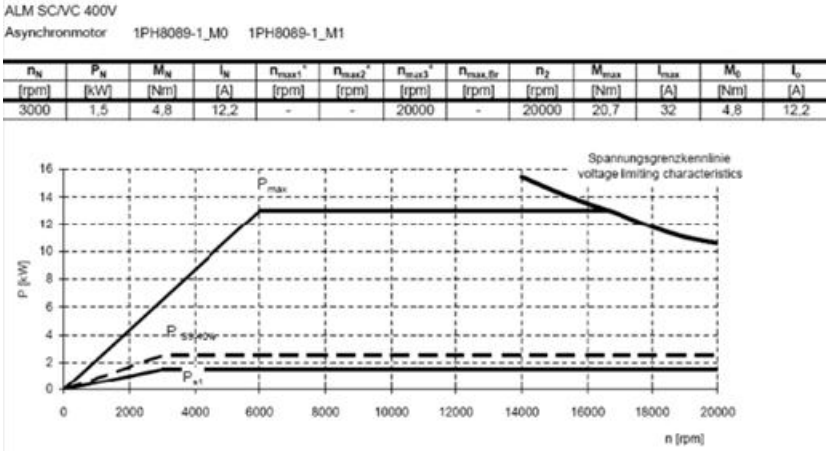


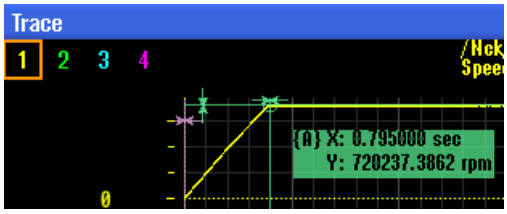

Figure 6-7 1PH8 induction motor

### Recording a new trace

Procedure:

1. Execute the Spindle and Trace function. The time required for acceleration to the speed setpoint can now be checked.

   Trace selection: Actual speed




2. Set MD35200 so that it meets the machine manufacturer's requirements for the spindle acceleration:

   MD35200[0] $MA_GEAR_STEP_SPEEDCTRL_ACCEL

   MD35200[1] $MA_GEAR_STEP_SPEEDCTRL_ACCEL

## 6.5.3 Spindle: Checking the position controller

### Testing the position controller

A test can now also be performed to set the acceleration and jerk of the position control. This test is performed by tapping without using a compensating chuck.

Procedure:

1. Check MD35550 $MA_DRILL_VELO_LIMIT[0] und [1]; the maximum velocity during tapping.
2. Check SD55484 $SCS_DRILL_TAPPING_SET_MC[0] = 1;, it holds the spindle in the positioning mode during tapping.

   The following program can be used to drill ten holes at the same position:

```
SUPA D0 G0 G90 Z-200

FFWON

M19

SOFT

LAB:

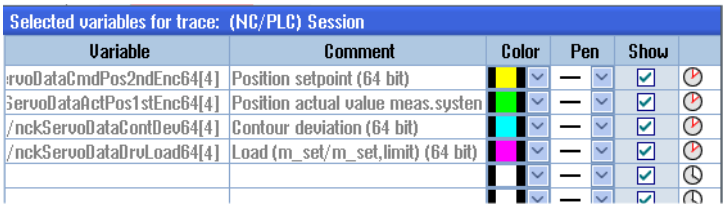
G331 Z-214 S4000 K1

G332 Z-200 S4000 K1

REPEAT LAB P=9

M30
```

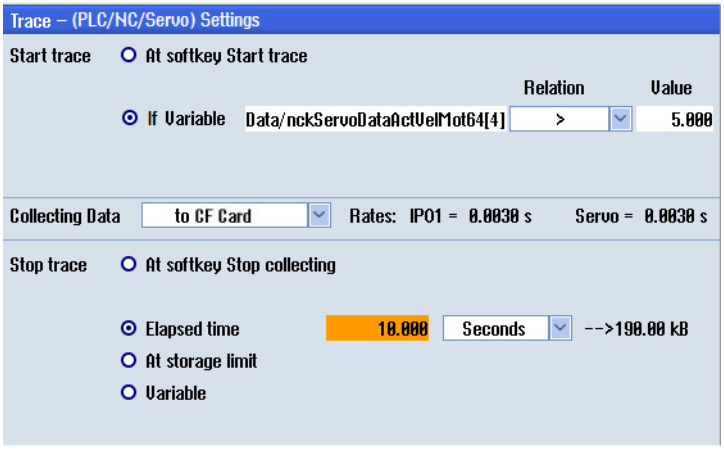
3. Select the following variables:
   - Position setpoint (64-bit)
   - Measuring system actual position (64-bit)
   - Contour deviation (64-bit)
   - Load (m_set/m_set_limit) (64-bit)




**Note**

Ensure that the correct signal is selected for the monitoring of measuring systems, e.g. measuring system 1 or 2.

4. Use the "Settings" softkey to select the trace trigger, etc. The example shows a trace that is initiated when the actual position exceeds five degrees. The trace is recorded for ten seconds.



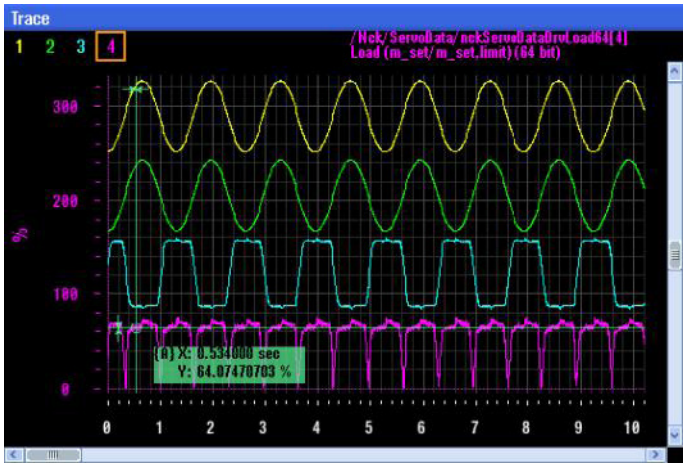

5. Start the trace and press <CYCLE START> on the machine control panel.

## Examples

Because the load reaches 64% in the following example, the position-control acceleration can be increased. Because the spindle can operate above the S1 curve during the tapping, it must be guaranteed that the load does not reach 100% and so no alarms for contour monitoring are issued.

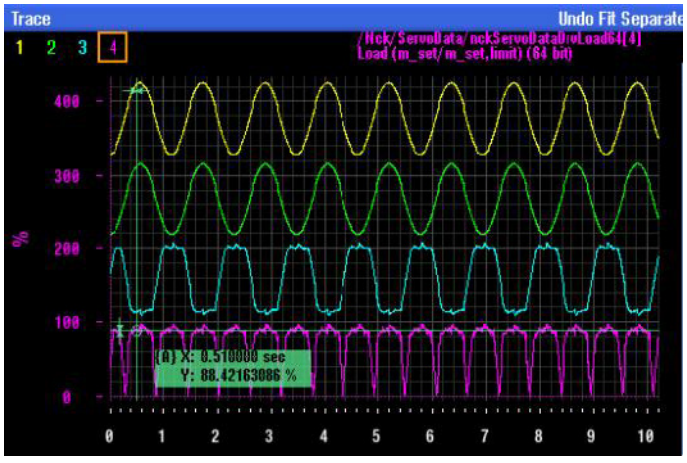
1. To achieve optimum results, optimize MD35210 $MA_GEAR_STEP_POSCTRL_ACCEL and MD32431 $MA_MAX_AX_JERK.
2. Also check the adaptable control range p1464, p1465 of the spindle.

**Spindle: Load 64%**




The following example shows the effect of increasing the position-control acceleration from 120 to 160; the load then increases from 64% to 88%. If the application involves a pure tapping machine, a test can be performed for which 100 holes are tapped. The trace recording is then repeated to ensure that 100% load is never reached.

**Spindle: Load 88%**



# 6.6 Circularity test

## 6.6.1 Circularity test: Function

The circularity test serves to set and assess the dynamic response for interpolating axes and to analyze the contour accuracy on the quadrant transitions (circular contours) achieved by means of friction compensation (conventional quadrant error compensation).

The circularity test is used to check the interpolation of the axes which work together. This function measures a circle with reference to the motor or to the direct measuring system. Alignment of the machine-mechanical equipment is not taken into account in the result. This gives the commissioning engineer the option of separating problems with the controller tuning from mechanical problems.

The following axis machine data and parameters are checked with this procedure:

- MD32200, MD32400, MD32402, MD32410, MD32490, MD32500, MD32510, MD32520, MD32540 MD32620, MD32640, MD32810, MD32900, MD32910, MD32930, MD32940
- p1421 bis p1426, p1400, p1433, p1434

The following position error compensations should be disabled if this procedure is carried out:

- MD32450 Backlash compensation
- MD32500 Friction compensation active
- MD32700 Encoder/lead error compensation
- MD32710 Enable sag compensation
- MD32750 Temperature compensation type

**Note**

MD32450 backlash must be adjusted via an external device, such as a circularity test or gauge.

**Example:**

NC test program for the measurement of the X-Y axes:

```
FFWON

SOFT

G90 G01 F3000 X400 Y200 Z500

LAB:

G91 G64 G02 X0 Z0 I10

GOTOB LAB

M30
```

Position, feedrate and active plane must be adjusted to the machine!

### Result

The best contour results are achieved when the circular form test results are in the correct actual size, shape and minimum p/p deviation between a combined interpolation of the axes (X-Y, X-Z, Y-Z).

An NC program in the MDA operating mode and the circularity test function are used to measure and evaluate these results. The "worst case" of a circle radius and the path velocity must lead to a realistic radial acceleration of which the machine is capable.

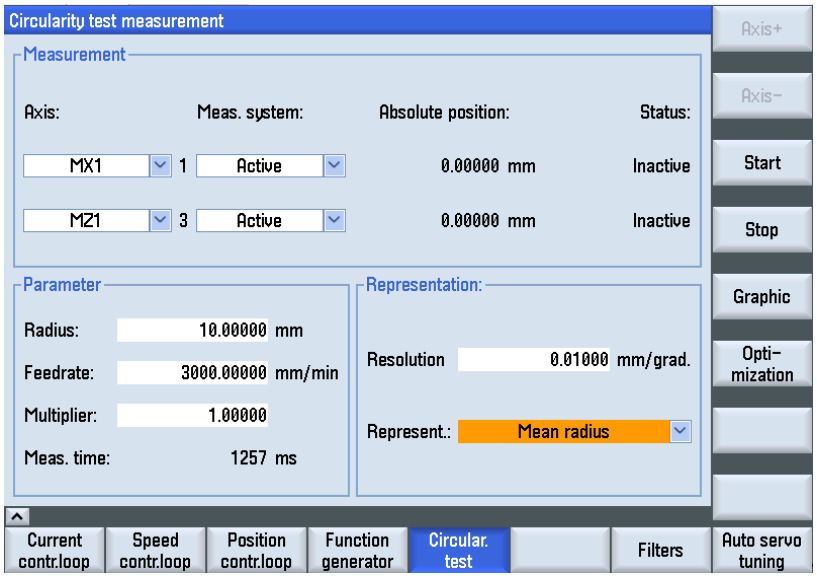
Machine manufacturers generally have specifications for the radius and feedrate test circuit.

Circularity tests of machine manufacturers usually use a radius of 100 mm or 150 mm with feedrate speeds that are determined by the machine manufacturer. The machine manufacturer determines the specifications for an acceptable result.

High-speed processes generally have higher requirements for testing circles with high-speed milling machines and can range from circle radii of 10 - 25 mm and feedrates of 5 - 10 m/min. For high-speed milling machines, the results are generally acceptable if the P/P deviation ≤ 0.010 mm and the actual size of the circle is equal to the programmed radius, and in the worst-case scenario to the path velocity.

## 6.6.2 Circularity test: Performing the measurement

### Setting parameters



Enter the following parameters to carry out a measurement:

- **"Measurement":** Selection of the two axes that are intended to be measured and selection of the measurement system. Parked encoders are not displayed for selection.
- **"Parameter":** The parameter settings in the "Radius" and "Feed" input fields must correspond to the values from the part program that controls the circular motion of the axes, taking account of the feed override switch setting.
- **"Representation":** Parameters for displaying the graphics
  - "Resolution" (scaling) of the diagram axes in [mm/scaled]
  - "Display" via the mean radius or programmed radius

## Perform measurement

Procedure:

1. In the "Commissioning" operating area select the "Optim./test" softkey → "Circularity test" softkey.
2. Select the axes that are intended to be measured with the <SELECT> key or with the "Axis+"/"Axis -" softkeys.
3. Set the parameters for the measurement: "Radius" and "Feed"

   The "Measurement time" display field shows the measurement time calculated from the "Radius" and "Feed" values for recording the actual position values during the circular movement:

   If the measurement time is not sufficient then only parts of the circle are portrayed. The measurement time can be increased by reducing the feed value. This also applies if the circularity test is started from the stationary condition.
4. Set the parameters for displaying the graphic:

   If the measurement time calculated from this exceeds the time range that can be displayed (maximum measurement time = position controller cycle frequency * 2048), an appropriate sample time is calculated for recording (n * position controller cycle frequency), so that a complete circle can be displayed.

Further actions:

- To start the measurement, press the "Start" softkey.
- To stop the measurement, press the "Stop" softkey.
- To make further adjustments for tuning, press the "Tuning" softkey.
- On a new softkey bar, you can navigate **directly** to the following areas:
  - "Service axis" in the "Diagnostics" operating area
  - "Axis machine data"
  - "Drive machine data"
  - "User Views"

- To save the parameters for a measurement, press the "Save parameter" softkey.
- To repeat a measurement with the same parameters, for example, press the "Load parameter" softkey.

### Displaying a graphic

To display the measurement result as a graphic, press the "Graphic" softkey.

## 6.6.3 Circularity test: Examples

The MD32400 $MC_AX_JERK_ENABLE axial jerk limitation is set via a time constant and is always active.

Machine data for setpoint filter:

- MD32402 $MC_AX_JERK_MODE = Type 2 is recommended, type 1 is preset for compatibility reasons. Parameterizing a pure band-stop filter is expressly not recommended.
- MD32402 $MA_AX_JERK_MODE (filter type) and MD32410 $MA_AX_JERK_TIME > 0 is effective only if MD32400 $MA_AX_JERK_ENABLE = 1 is set.

### Example 1 for optimization

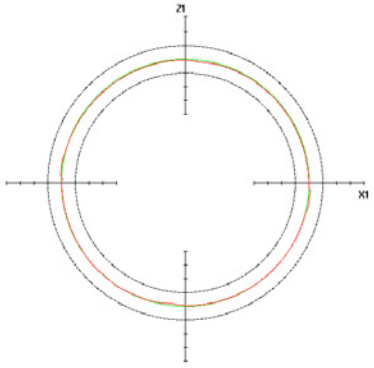
Machine data after optimization of the axes:

| Parameters / machine data | | X axis | Z axis |
|---|---|---|---|
| MD32200 | $MC_POSCTRL_GAIN | 8.500 | 8.500 |
| p1460 | SPEEDCTRL_GAIN1 | 3.01 | 3.89 |
| p1462 | SPEEDCTRL_INTEGRATOR_TIME_1 | 6.18 | 6.18 |
| p1463 | SPEEDCTRL_REF_MODEL_FREQ | 106.3 | 106.3 |
| p1440 | NUM_SPEED_FILTERS | 0 | 0 |
| MD32610 | $MC_VELO_FFW_WEIGHT | 1.0 | 1.0 |
| MD32620 | $MC_FFW_MODE | 4 | 4 |
| MD32810 | $MC_EQUIV_SPEEDCTRL_TIME | 0.0022 | 0.0022 |
| MD32400 | $MC_AX_JERK_ENABLE | 0 | 0 |

The actual value of the radius is generally too large with optimized feed precontrol. This can be corrected with the MD32410 $MC_AX_JERK_TIME time constant. Use a time constant in all axes if required.

This circle displays the results after the feed precontrol has been optimized. However, the mean radius is 0,0019 mm too large:

X1: active measuring system
Z1: active measuring system




**Parameter**

| | |
|---|---|
| Radius: | 10.00000 mm |
| Feedrate: | 3000.00000 mm/min |
| Measurement time: | 1257 ms |
| X1: | active measuring system |
| Z1: | active measuring system |

**Representation**

| | |
|---|---|
| Resolution: | 0.01000 mm |
| Display of. | mean radius |
| Radius: | 10.00190 mm |
| Delta R: | 4.02698 µm |

| Parameters / machine data | | X axis | Z axis |
|---|---|---|---|
| MD32200 | $MC_POSCTRL_GAIN | 8.500 | 8.500 |
| p1460 | SPEEDCTRL_GAIN1 | 3.01 | 3.89 |
| p1462 | SPEEDCTRL_INTEGRATOR_TIME_1 | 6.18 | 6.18 |
| p1463 | SPEEDCTRL_REF_MODEL_FREQ | 106.3 | 106.3 |
| p1440 | NUM_SPEED_FILTERS | 0 | 0 |
| MD32610 | $MC_VELO_FFW_WEIGHT | 1.0 | 1.0 |
| MD32620 | $MC_FFW_MODE | 3 | 3 |
| MD32810 | $MC_EQUIV_SPEEDCTRL_TIME | 0.0022 | 0.0022 |
| MD32400 | $MC_AX_JERK_ENABLE | 1 | 1 |
| MD32402 | $MC_AX_JERK_MODE | 2 | 2 |
| MD32410 | $MC_AX_JERK_TIME | **0.012** | **0.012** |

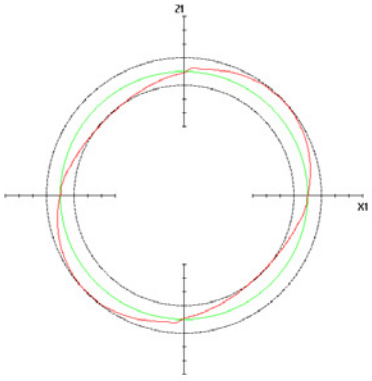
## Example 2 for optimization

This circle shows the effect of a slightly different time constant for the axial jerk filter. The time constant is adapted in order to correct this type of error:

| Parameters / machine data | | X axis | Z axis |
|---|---|---|---|
| MD32400 | $MC_AX_JERK_ENABLE | 1 | 1 |
| MD32402 | $MC_AX_JERK_MODE | 2 | 2 |
| MD32410 | $MC_AX_JERK_TIME | **0.012** | **0.0125** |

X1: active measuring system

Z1: active measuring system




**Parameter**

| | |
|---|---|
| Radius: | 10.00000 mm |
| Feedrate: | 3000.00000 mm/min |
| Measurement time: | 1257 ms |
| X1: | active measuring system |
| Z1: | active measuring system |

**Representation**

| | |
|---|---|
| Resolution: | 0.01000 mm |
| Representation: | mean radius |
| Radius: | 10.00029 mm |
| Delta R: | 25.47002 µm |

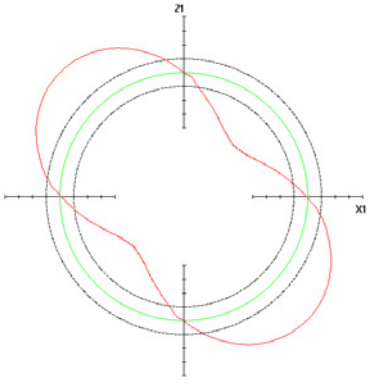
### Example 3 for optimization

This circle shows the effect of a considerably different time constant for the axial jerk filter. The time constant is adapted in order to correct this type of error:

| Parameters / machine data | | X axis | Z axis |
|---|---|---|---|
| MD32400 | $MC_AX_JERK_ENABLE | 1 | 1 |
| MD32402 | $MC_AX_JERK_MODE | 2 | 2 |
| MD32410 | $MC_AX_JERK_TIME | **0.015** | **0.012** |

X1: active measuring system

Z1: active measuring system




| **Parameter** | |
|---|---|
| Radius: | 10.00000 mm |
| Feedrate: | 3000.00000 mm/min |
| Measurement time: | 1257 ms |
| X1: | active measuring system |
| Z1: | active measuring system |
| **Representation** | |
| Resolution: | 0.01000 mm |
| Representation: | mean radius |
| Radius: | 9.98971 mm |
| Delta R: | 75.67665 µm |

## 6.6.4 Circularity test: Saving data

The following data can be backed up with the circularity test:

- **Save parameters:** The parameters entered can be saved as a file.

  The following path is the default path:
  user/sinumerik/hmi/log/optimization/cicular/<name>.sup

- **Save graphic:** If the graphic has been saved correctly the message "Data was saved" appears, otherwise an "Error in saving the file" error message is received.

  The following path is the default path:
  user/sinumerik/hmi/log/optimization/cicular/<name>.sud

- **Print graphic:** The graphic is saved as a pixel graphic in PNG format. Type in a name. The name can be freely selected. The following path is the default path:
  user/sinumerik/hmi/log/optimization/cicular/<name>.png

## Save parameters

The file format is structured as follows (with comments in brackets[ ])

```
H: CstPar [Identifier for circularity test parameters]
V: 5.0 [Version number of the file format]
@Measurement
P 1: 1 [Axis number first axis]
P 2: 2 [Axis number second axis]
P 3: 0 [Measuring system Axis 1 - 0: active; 1:first; 2: second]
P 4: 0 [Measuring system Axis 2 - 0: active; 1:first; 2: second]
@ parameters
P 10: 30  [Radius]
P 11: 3000 [Feed]
P 12: 3770 [Measuring time]
P 14: 1 [Multiplier]
@Representation
P 20: 10  [Resolution]
P 21: 8 [mean / prog. radius - 8 = mean R ; 9 = prog. Radius]
```

## Save graphic

The file format is structured as follows (with comments in brackets[ ]):

```
H: CstPic [Identifier for circularity test graphic]
V: 5.0 [Version number of the file format]
@ parameters
P 1: 30 [Radius]
P 2: 3000 [Feed]
P 3: 3770 [Measuring time]
P 4: 0 [Measuring system Axis 1 - 0: active; 1:first; 2: second]
P 5: 0 [Measuring system Axis 2 - 0: active; 1:first; 2: second]
@Representation
P 10: 10 [Resolution]
P 11: 9 [mean / prog. radius - 8 = mean R ; 9 = prog. Radius]
P 12: X1 [Axis name 1]
P 13: Z1 [Axis name 2]
@Intermediate values
P 20: 15.6632 [max. radius measured values]
```

```
P 21: 10.9326 [min. radius measured values]

P 22: 13.6694 [mean radius measured values]

P 23: 1886 [Number of measured values]

@Additional values

P 30: 1000 [Accuracy (1/P30))

@Physical units

P 40: 5370 [Text number radius unit]

P 41: 5381 [Text number feed unit]

P 42: 6165 [Text number Resolution unit]

P 43: 5346 [Text number DeltaRadius unit]

P 44: 0 [New: Operate: Basislengthunit]

@Abscissa

Ai:   [Abscissa values i: 0..P23]

@Ordinate

Oi:   [Ordinate values i: 0..P23]

@Radius

Ri:   [Radius values i: 0..P23]
```

# Service Planner

# 7

## Overview

Using the service planner dialog at the user interface or programming tool, time intervals and alarm sequences for the tasks to be processed (mainly machine maintenance tasks) can be edited, started, deactivated or reactivated.

The numeric data of the tasks is organized in data blocks and provided in the user interface for the HMI, PLC user program and programming tool. The designations of the particular task are managed and edited by the operating software and displayed together with the numerical data.

The PLC firmware accesses the user interface data blocks, processes the data and then provides the results in the form of remaining times as well as warnings and alarms in data blocks. The Service Planner is processed every minute in the PLC firmware. When the control is switched off, the actual data of the maintenance tasks is frozen. When the control is switched on, processing is continued with these retentively saved values.

The PLC user program evaluates the actual data and generates warning and alarm messages in numeric form either with or without Power OFF status. The alarm handler converts these messages with the appropriate PLC alarm text file oem_alarm_plc_<lng>.ts to a message for the operator which is displayed on the user interface (<lng> currently set language) and which can be logged, if required.

## Interfaces and configuration

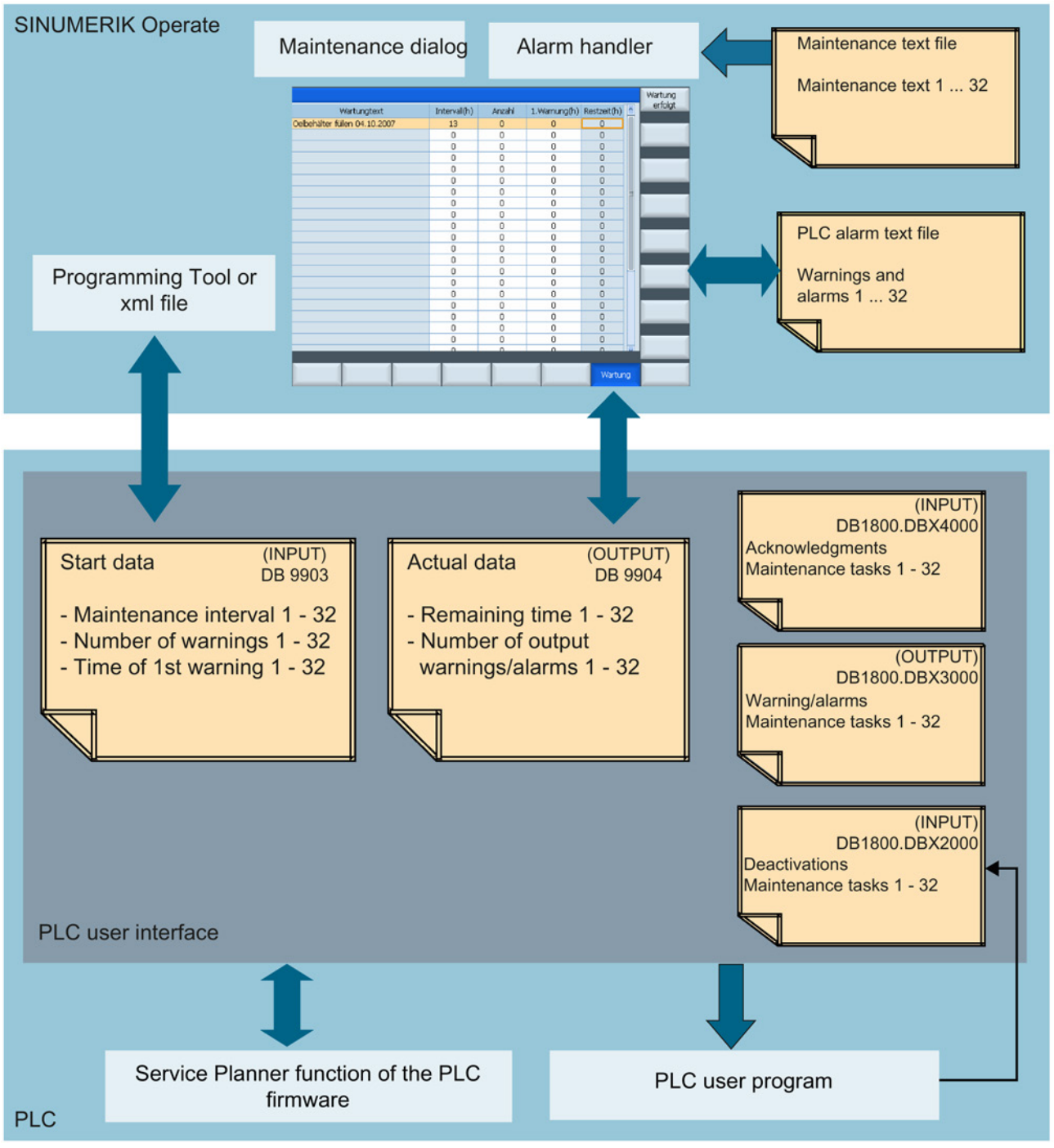



Figure 7-1 Service Planner - interfaces

# 7.1 Interfaces in the PLC user program

## PLC user program

The PLC user program operates the user interface. This includes, in particular, the bit interfaces of the DB1800 and the evaluation of the remaining time in DB9904. You must ensure that appropriate messages are displayed for the warnings and alarms.

During the configuration of these messages, so-called alarm responses can be selected, e.g. the machine is "disabled" after a Power OFF message.

The alarm messages are configured after specification of the SINUMERIK 828D PLC messages. The texts are entered with the alarm text editor. Then the texts are available for the operating software:

- Evaluation of the alarm data and actual data with the aim of generating PLC warnings and PLC alarms. Additional signals can be included in the evaluation logic.
- Optional linking of the deactivation bits with bit memory or I/O signals.

**Note**

A sample program is supplied in the PLC function library. This can be adapted by machine manufacturers to meet their requirements.

## Interfaces to the PLC

The following data areas are available in the user interface:

| Data block | Meaning |
|---|---|
| DB9903 | Initial data |
| DB9904 | Actual data |
| DB1800.DBB2000 | Deactivate tasks |
| DB1800.DBB3000 | Alarms |
| DB1800.DBB4000 | Acknowledgments |
| DB1800.DBB5000 | Acknowledgment block |

## DB9903: Initial data

| DB9903 | Initial data table [r16] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBW0 | Interval 1 [h] | | | | | | | |
| DBW2 | Time of first warning 1 [h] | | | | | | | |
| DBW4 | Number of warnings to be output 1 | | | | | | | |
| DBW6 | Reserved 1 | | | | | | | |
| DBW8 | Interval 2 [h] | | | | | | | |
| DBW10 | Time of first warning 2 [h] | | | | | | | |

| DB9903 | Initial data table [r16] |
|---|---|
| DBW12 | Number of warnings to be output 2 |
| DBW14 | Reserved 2 |
| ... | ... |
| DBW248 | Interval 32 [h] |
| DBW250 | Time of first warning 32 [h] |
| DBW252 | Number of warnings to be output 32 |
| DBW254 | Reserved 32 |

Interval: Number of hours after which the maintenance must be performed. When this time expires, the warning or alarm bit belonging to the task is set for the last time.

Time of the first warning: Number of hours after which the first warning is output. This time must be greater than or equal to the interval.

Number of warnings to be output: Number of n warnings to be output before the alarm. The alarm bit is therefore set maximum (n+1)-times, i.e n-times as warning and 1-time as alarm.

Reserved: Reserved for expansions.

**Example:**

Interval = 100

Time of the 1st warning = 80

Number of warnings to be output = 2

After the task is started, the warning/alarm bit is output for the first time after 80 hours, a second time after a further 10 hours (i.e. after a total of 90 hours), and the warning/alarm bit is set for the last time after 100 hours.

## DB9904: Actual data

| DB9904 | Actual data table [r16] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBW0 | Remaining time 1 [h] | | | | | | | |
| DBW2 | Number of warnings output 1 | | | | | | | |
| DBW4 | Reserved_1 1 | | | | | | | |
| DBW6 | Reserved_2 1 | | | | | | | |
| DBW8 | Remaining time 2 [h] | | | | | | | |
| DBW10 | Number of warnings output 2 | | | | | | | |
| DBW12 | Reserved_1 2 | | | | | | | |
| DBW14 | Reserved_2 2 | | | | | | | |
| ... | ... | | | | | | | |
| DBW248 | Remaining time 32 [h] | | | | | | | |
| DBW250 | Number of warnings output 32 | | | | | | | |

| DB9904 | Actual data table [r16] |
|---|---|
| DBW252 | Reserved_1 32 |
| DBW254 | Reserved_2 32 |

Remaining time: Number of hours remaining after the start of the task until it expires.

Remaining time ≠ 0 and associated alarm bit set: Warning

Remaining time = 0 and associated alarm bit set: Alarm

Number of warnings output: Number n of warnings that have already been output. If the interval has expired completely, the output value is (n+1):

n = "number of warnings to be output"

1 = alarm at the end of the interval

Reserved_1, ~_2: Reserved for expansions.

**Example:**

Interval = 100, time of the 1st warning = 80, number of warnings to be output = 2

After the task is started, the remaining time is decremented every hour.

- After 80 hours, the remaining time is 20 hours and the number of warnings that have been output is increased from 0 to 1.
- After a further 10 hours (i.e. a total of 90 hours), the remaining time is 10 hours and the number of warnings that have been output is increased from 1 to 2.
- After 100 hours, the remaining time is 0 and the number of warnings that have been output is 3 (= 2 warnings plus 1 alarm).

## DB1800: Acknowledgments

| DB1800 | Acknowledgments [r/w] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBB4000 | Acknowl-edgment 8 | Acknowl-edgment 7 | Acknowl-edgment 6 | Acknowl-edgment 5 | Acknowl-edgment 4 | Acknowl-edgment 3 | Acknowl-edgment 2 | Acknowl-edgment 1 |
| DBB4001 | Acknowl-edgment 16 | Acknowl-edgment 15 | Acknowl-edgment 14 | Acknowl-edgment 13 | Acknowl-edgment 12 | Acknowl-edgment 11 | Acknowl-edgment 10 | Acknowl-edgment 9 |
| DBB4002 | Acknowl-edgment 24 | Acknowl-edgment 23 | Acknowl-edgment 22 | Acknowl-edgment 21 | Acknowl-edgment 20 | Acknowl-edgment 19 | Acknowl-edgment 18 | Acknowl-edgment 17 |
| DBB4003 | Acknowl-edgment 32 | Acknowl-edgment 31 | Acknowl-edgment 30 | Acknowl-edgment 29 | Acknowl-edgment 28 | Acknowl-edgment 27 | Acknowl-edgment 26 | Acknowl-edgment 25 |

Acknowledgment n: The acknowledgment bit assigned to task n:

Under the precondition that the corresponding acknowledgment blocking bit is not set, the setting of the acknowledgment bit restarts the task and, in particular, the actual data of the task is set:

Remaining time = interval

Number of warnings output = 0

The bit is automatically reset at the end of the PLC cycle.

**Example:**

Interval = 100, time of the 1st warning = 80, number of warnings to be output = 2

After the setting of the associated acknowledgment bit, the remaining time is set to the interval time and the number of output warnings is zero - assuming the associated acknowledgment blocking bit is not set.

## DB1800: Alarms

| DB1800 | Warnings/Alarms [r] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBB3000 | Alarm 8 | Alarm 7 | Alarm 6 | Alarm 5 | Alarm 4 | Alarm 3 | Alarm 2 | Alarm 1 |
| DBB3001 | Alarm 16 | Alarm 15 | Alarm 14 | Alarm 13 | Alarm 12 | Alarm 11 | Alarm 10 | Alarm 9 |
| DBB3002 | Alarm 24 | Alarm 23 | Alarm 22 | Alarm 21 | Alarm 20 | Alarm 19 | Alarm 18 | Alarm 17 |
| DBB3003 | Alarm 32 | Alarm 31 | Alarm 30 | Alarm 29 | Alarm 28 | Alarm 27 | Alarm 26 | Alarm 25 |

Alarm n: The alarm bit assigned to task n.

The bit is set for one PLC cycle each as warning (remaining time ≠ 0) and as alarm (remaining time = 0).

## DB1800: Deactivate tasks

| DB1800 | Task deactivation [r/w] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBB2000 | Deactivation 8 | Deactivation 7 | Deactivation 6 | Deactivation 5 | Deactivation 4 | Deactivation 3 | Deactivation 2 | Deactivation 1 |
| DBB2001 | Deactivation 16 | Deactivation 15 | Deactivation 14 | Deactivation 13 | Deactivation 12 | Deactivation 11 | Deactivation 10 | Deactivation 9 |
| DBB2002 | Deactivation 24 | Deactivation 23 | Deactivation 22 | Deactivation 21 | Deactivation 20 | Deactivation 19 | Deactivation 18 | Deactivation 17 |
| DBB2003 | Deactivation 32 | Deactivation 31 | Deactivation 30 | Deactivation 29 | Deactivation 28 | Deactivation 27 | Deactivation 26 | Deactivation 25 |

Deactivation n: The deactivation bit assigned to task n.

If the bit is set via SINUMERIK Operate or from the PLC user program, the current state of task n is frozen and no longer processed.

**TRUE:** Task deactivated

**FALSE:** Task active

This means it is possible, for example, to adapt the maintenance interval according to the actual runtime of the modules.

## DB1800: Acknowledgment block

| DB1800 | Acknowledgment block [r/w] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBB5000 | Acknowl-edgment block 8 | Acknowl-edgment block 7 | Acknowl-edgment block 6 | Acknowl-edgment block 5 | Acknowl-edgment block 4 | Acknowl-edgment block 3 | Acknowl-edgment block 2 | Acknowl-edgment block 1 |
| DBB5001 | Acknowl-edgment block 16 | Acknowl-edgment block 15 | Acknowl-edgment block 14 | Acknowl-edgment block 13 | Acknowl-edgment block 12 | Acknowl-edgment block 11 | Acknowl-edgment block 10 | Acknowl-edgment block 9 |
| DBB5002 | Acknowl-edgment block 24 | Acknowl-edgment block 23 | Acknowl-edgment block 22 | Acknowl-edgment block 21 | Acknowl-edgment block 20 | Acknowl-edgment block 19 | Acknowl-edgment block 18 | Acknowl-edgment block 17 |
| DBB5003 | Acknowl-edgment block 32 | Acknowl-edgment block 31 | Acknowl-edgment block 30 | Acknowl-edgment block 29 | Acknowl-edgment block 28 | Acknowl-edgment block 27 | Acknowl-edgment block 26 | Acknowl-edgment block 25 |

Acknowledgment block n: The acknowledgment blocking bit assigned to task n.

If the bit is set via SINUMERIK Operate or from the PLC user program, the task is not acknowledged even when the acknowledgment bit is set.

**TRUE:** Acknowledgment of the task blocked

**FALSE:** Acknowledgment of the task permitted

In this way, it is possible, for example, to incorporate a sensor in the PLC user program, which signals that the maintenance task has been performed, and, if required, block the acknowledgment.

# 7.2 Interfaces to the operating software

## Overview

To process these service/maintenance tasks, you have the following options:

- Managing the data in the PLC user program using the programming tool.
- Maintenance tasks are created via an XML interface.

Using this segmentation, inconsistencies between the PLC block and the XML scripts are avoided if the PLC block is changed in the PLC user program via the programming tool or as a result of a commissioning archive.

## Configuration

You can configure up to 32 maintenance tasks. The following columns are shown in the configuring mode:

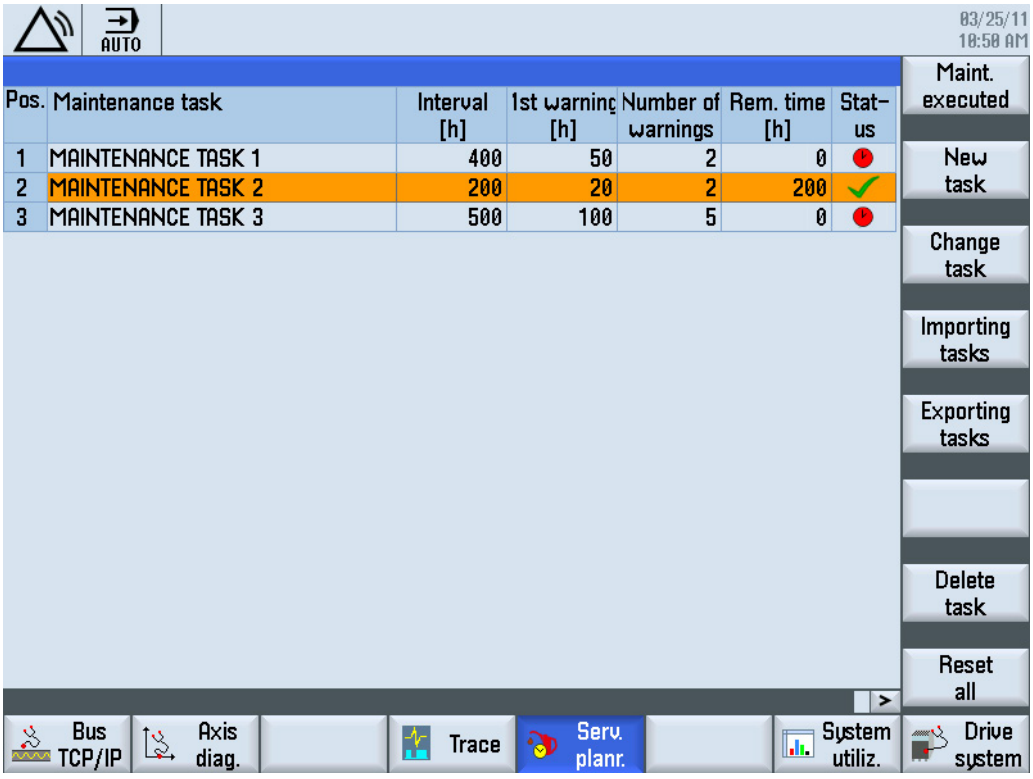
| Designation of the column | Meaning |
|---|---|
| Maintenance task | Name of the maintenance task. |
| Interval [h] | Maximum time until the next maintenance in hours; if this value ≠ 0, this data set is accepted by the PLC as a valid maintenance task. |
| 1st warning [h] | Time in hours after which the first warning is displayed; this value must be less than that of the interval. |
| Number of warnings | Number of warnings that are output by the PLC before the PLC sets the alarm bit for the last time after the interval has expired (remaining time == 0). |
| Remaining time [h] | Time until the interval expires in hours. |
| Status | • A green check mark indicates that the time up to the next maintenance is still running.<br>• A red clock symbol indicates that there is a maintenance task to be performed. |

The dialog is called with different contents depending on the access level.

## Configuration mode

Access level 2: Service

Maintenance tasks can be created, changed and also deleted in this mode. The maintenance tasks can also be acknowledged. All columns are visible. Navigation between the columns is with <Tab> or <Key Left/Right>.




**Note**

**Number assignment**

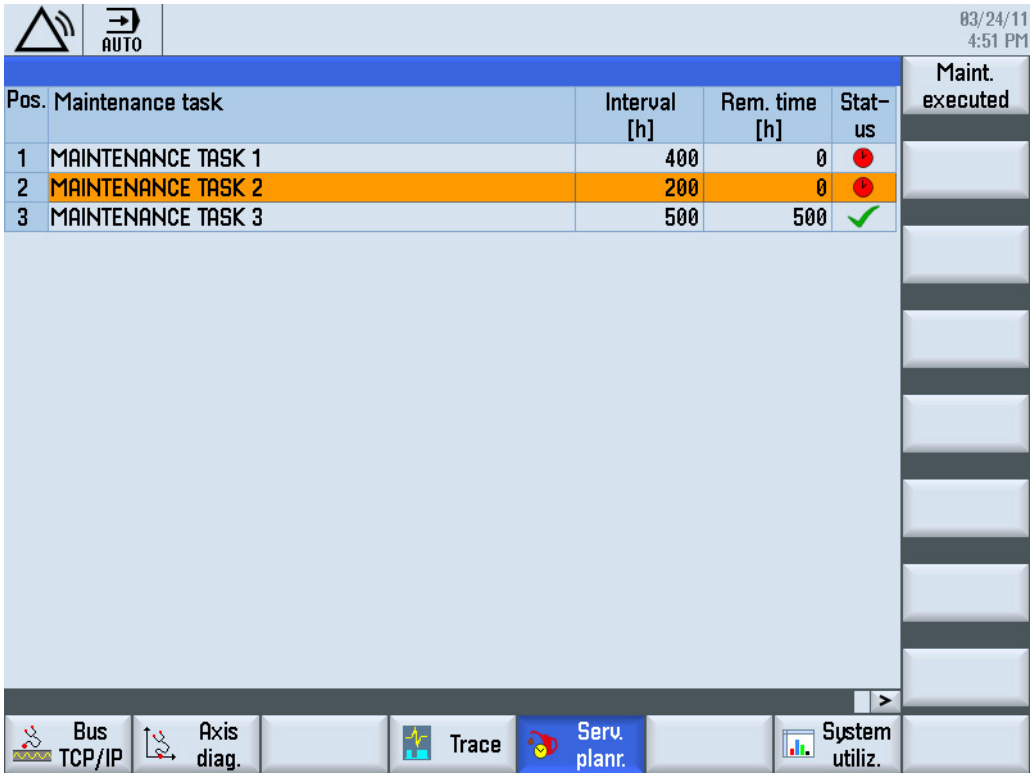
A number is automatically assigned when a task is created. If this is not desirable, then the tasks must be configured with the corresponding numbers.

This is recommended, for example, if task m is deleted by mistake and this has to be created again under the same number because of the evaluation in the PLC user program.

## Standard mode

Access level 3: User

In the standard mode, the actual status of the maintenance tasks is displayed. The following columns are visible, but cannot be edited: "Maintenance task", "Interval", "Remaining time" and "Status".



# 7.3 This is how you import and export maintenance tasks

## Overview

In the dialog, there are two soft keys that can be used to read-in and read-out the configuration files with the maintenance tasks:

- Importing maintenance tasks
- Exporting maintenance tasks

## Configuring maintenance tasks

Procedure:

1. Generate a configuration file using the structure described below.

   The filename in lower-case letters can be freely selected.

2. As a minimum, generate the oem_maintenance_<lng>.ts, file, which contains the language-dependent designations for the maintenance tasks.
3. Copy the two files to a storage medium (CompactFlash card or USB FlashDrive).
4. Insert the storage medium into the appropriate slot at the front.
5. Press the "Import maintenance tasks" softkey.
6. Confirm with "OK".

   The files are copied. The language-dependent tasks are immediately visible.

Press the "Export maintenance tasks" softkey in order to make changes or supplements. This means that you copy these files to a storage medium, which you can then edit using an external editor.

## Structure of the xml configuration file

The following identifiers are permissible:

| Identifier | Meaning |
|---|---|
| <MAINTENANCE_TASK> | Main Tag |
| <TASK_ID> | Number of the maintenance task |
| <INTERVALL> | Maintenance interval time |
| <FIRST_WARNING> | Time interval until the first alarm is output. |
| <NUMBER_OF_WARNING> | Number of alarms |

The file name can be freely selected, for example task.xml.

**task.xml**

```
<MAINTENANCE>
  <MAINTENANCE_TASK>
    <TASK_ID>0</TASK_ID>
    <INTERVALL>3</INTERVALL>
    <FIRST_WARNING>1</FIRST_WARNING>
    <NUMBER_OF_WARNING>1</NUMBER_OF_WARNING>
  </MAINTENANCE_TASK>
  <MAINTENANCE_TASK>
    <TASK_ID>1</TASK_ID>
    <INTERVALL>3</INTERVALL>
    <FIRST_WARNING>2</FIRST_WARNING>
    <NUMBER_OF_WARNING>1</NUMBER_OF_WARNING>
  </MAINTENANCE_TASK>
  ...
  <MAINTENANCE_TASK>
    <TASK_ID>2</TASK_ID>
    <INTERVALL>3</INTERVALL>
    <FIRST_WARNING>2</FIRST_WARNING>
    <NUMBER_OF_WARNING>1</NUMBER_OF_WARNING>
  </MAINTENANCE_TASK
</MAINTENANCE>
```

## Structure of the oem_maintenance_<lng>.ts file

This file has the extension ".ts" and contains all the language-dependent warning texts that were entered in the dialog. This file is available in the binary format (*.qm) after the system powers-up the next time.

Target directory: `/oem/sinumerik/hmi/lng`

**oem_maintenance_<lng>.ts**

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE TS>
<TS>
  <context>
    <name>maintenance</name>
    <message>
      <source>1</source>
      <translation>Maintenance task 1</translation>
      <chars>44</chars>
    </message>
    <message>
```

**oem_maintenance_<lng>.ts**

```
      <source>2</source>
      <translation>Maintenance task 2</translation>
      <chars>44</chars>
    </message>
. . .
    <message>
      <source>32</source>
      <translation>Maintenance task 32</translation>
      <chars>44</chars>
    </message>
  </context>
</TS>
```

## Editing text entries

The maintenance texts are entered in the dialog with the values for the interval, time of the first warning and the number of warnings. In addition, the .ts file can also be edited in the alarm text editor if the necessary entry is available in the configuration file "oem_alarms_config.xml".

## Integration in the existing language concept

When started, the Service Planner reads the oem_maintenance_<lng>.ts file with the language set in the language selection menu. If this is not available, the English version is read which has to be available for the commissioning.

# 7.4 Acknowledging maintenance tasks

## Acknowledging maintenance tasks

After completion of the maintenance activities, the maintenance tasks are acknowledged by the PLC user program or in the dialog of the operating software using the "Maintenance completed" softkey.

The acknowledgment bit for the maintenance task is set by making the acknowledgment in the dialog, the PLC then deletes "Number of output warnings/alarms" in the actual data and loads the interval value with the remaining time. This tells the user that the maintenance task has been successfully acknowledged.

The acknowledgment point in time of a maintenance task has the following effect:

- **Acknowledgment before the interval has expired**

  The maintenance interval can be acknowledged at any time. Premature acknowledgment means the premature start of a new maintenance interval.

- **Acknowledgment after the interval has expired**

  Acknowledgment of the maintenance interval restarts the task.

---

**Note**

**Access level for acknowledgment**

The access level to acknowledge a maintenance task is defined by the following machine data: MD51235 $MNS_ACCESS_RESET_SERV_PLANNER

Default setting: Access level 2 "service"

---

# Easy Extend

# 8

## 8.1 Function of Easy Extend

### Objective

Easy Extend provides you with a simple facility for commissioning, activating, deactivating or testing optional equipment. The available equipment and device states are displayed in a list by the control system. The system can manage a maximum of 64 devices. Softkeys are used to activate or deactivate a device.

The Easy Extend function is available in the operating area "Parameter" → "Extension menu" → "Easy Extend".

### Configuration

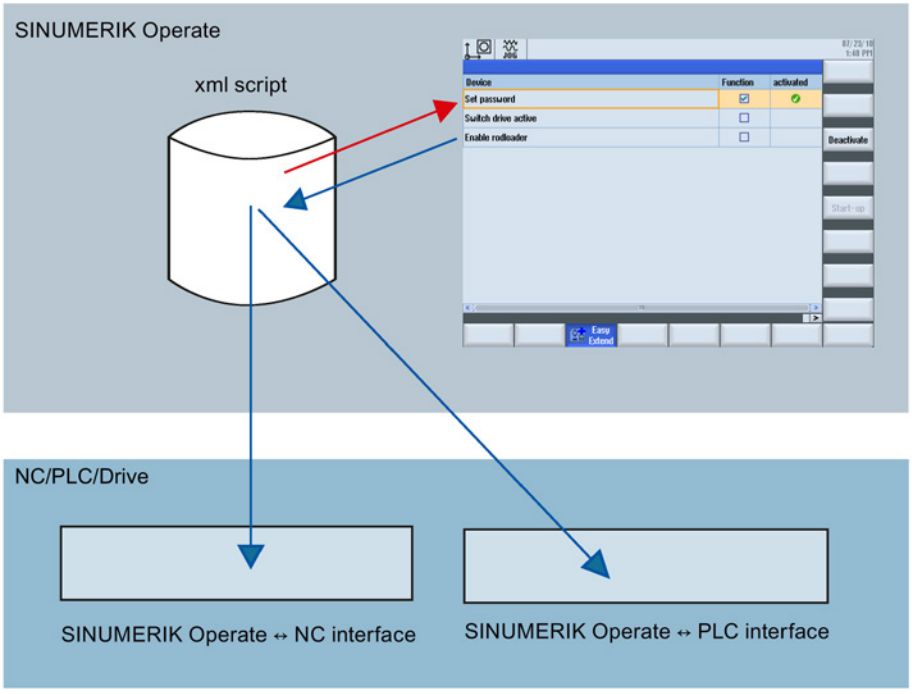



Figure 8-1 Mode of operation of Easy Extend

To use Easy Extend, the following functions should be configured by the machine manufacturer:

- **PLC ↔ SINUMERIK Operate interface**

  The optional devices are managed via the interface between the user interface and the PLC.

- **Script processing**

  The machine manufacturer saves the sequences to be executed for commissioning, activating, deactivating or testing a device, in a statement script.

- **Parameter dialog (optional)**

  The parameter dialog shows device information that is saved in the script file.

## Storage of the files

The Easy Extend files are stored on the system CompactFlash card in the "oem" (MANUFACTURER) and "oem_i" (INDIVIDUAL) directories.

| File | Name | Target directory |
|---|---|---|
| Text file | oem_aggregate_xxx.ts | /oem/sinumerik/hmi/lng/<br>/oem_i/sinumerik/hmi/lng/ |
| Script file | agm.xml | /oem/sinumerik/hmi/dvm<br>/oem_i/sinumerik/hmi/dvm |
| Archive file | Any | /oem/sinumerik/hmi/dvm/archives<br>/oem_i/sinumerik/hmi/dvm/archives |
| PLC user program | Any | PLC |

## References

The command scope of the script language is identical to Easy XML, and is described in the Programming Manual: SINUMERIK 840D sl/828D Easy XML

# 8.2 Configuration in the PLC user program

## Loading configurations

The configurations created are transferred to the manufacturer directory of the control, with the script and text file. Additionally, the corresponding PLC user program should be loaded.

## Programming the equipment

Communication between the operator component and the PLC takes place in the PLC user program via **data block DB9905**, in which 128 words are reserved to manage a maximum of 64 devices.

Four bytes with the following meanings are used for each device:

| Byte | Bit | Description | |
|---|---|---|---|
| 0 | 0 | == 1 | Device has been started up (HMI acknowledgment) |
| | 1 | == 1 | Device is to be activated (HMI request) |
| | 2 | == 1 | Device is to be deactivated (HMI request) |
| | 3-7 | Reserved | |
| 1 | 0-7 | Reserved | |
| 2 | 0 | == 1 | Device is active (PLC acknowledgment) |
| | 1 | == 1 | Device has an error |
| | 2-7 | Reserved | |
| 3 | 0-7 | Unique identifier for the device | |

## Adding axes

If the machine is extended with machine axes, it is important to install the drive objects (DO) in a fixed sequence because the start-up archive contains the constellation of the machine manufacturer's reference machine and cannot be applied if the sequence is changed.

It is recommended that the following settings be selected for the "control components":

- NC data
- PLC data
- Drive data
  - ACX format (binary)

**Note**

**Commissioning archive**

- In order to be able to use a commissioning archive in the Easy Extend script, these archives must be created **without** HMI data!

**Changing the machine configuration**

- Should there be any need to edit the drive machine data, this should be adapted in the control first. This procedure should be repeated for all devices and constellations.

## See also

This is how you create a commissioning archive (Page 342)

# 8.3 Option bits for machine manufacturers and dealers

## Distribution of the option bits

The option bits are managed in blocks of 16 bits. According to the default setting, block 3 is reserved for dealers. This means that machine manufacturers have 48 options and dealers have 16 options.

The Easy Extend function uses the following machine data to manage options:

**MD14510 $MN_USER_DATA_INT[0] to MD14510 $MN_USER_DATA_INT[3]**

This assignment of the addresses can also be redefined in the script using the identifier **OPTION_MD** so that dealers can specify their own address area. If the dealer wishes to offer more than 16 options for this machine type, then the blocks not used should be determined. This must be coordinated with the machine manufacturer.

Block 3 must be programmed in the script using operations to communicate the dealer area. Further, a fixed device index should be assigned for every device.

**Example to redefine any area:**

```
<option_md name="address identifier of the data" index="<index>"/>
  index - identifier of the area index:
    0: Device 0 to 16 (default setting)
    1: Device 17 to 32
    2: Device 33 to 48
    3: Device 49 to 64
```

## Mapping the device on the PLC interface

Each device must be assigned a unique index in order to permanently assign devices an option bit and to be able to access the PLC interface. The areas are permanently assigned using the attribute **option_bit**. If an assignment is not made, then the bits and the interface are defined by the index, which is assigned to the device in the following list:

| Index | MD14510 | Data block | Device designation |
|---|---|---|---|
| 0 | $MN_USER_DATA_INT[0] Bit 0 | DB9905.DBB0 | Device 1 |
| 1 | $MN_USER_DATA_INT[0] Bit 1 | DB9905.DBB4 | Device 2 |
| 2 | $MN_USER_DATA_INT[0] Bit 2 | DB9905.DBB8 | Device 3 |
| 3 | $MN_USER_DATA_INT[0] Bit 3 | DB9905.DBB12 | Device 4 |
| .... | ... | ... | ... |
| 47 | $MN_USER_DATA_INT[2] Bit 15 | DB9905.DBB188 | Device 48 |
| 48 | $MN_USER_DATA_INT[3] Bit 0 | DB9905.DBB192 | Device 49 |
| 49 | $MN_USER_DATA_INT[3] Bit 1 | DB9905.DBB196 | Device 50 |
| .... | ... | ... | ... |

| Index | MD14510 | Data block | Device designation |
|---|---|---|---|
| 62 | $MN_USER_DATA_INT[3] Bit 14 | DB9905.DBB248 | Device 63 |
| 63 | $MN_USER_DATA_INT[3] Bit 15 | DB9905.DBB252 | Device 64 |

**Example of a fixed assignment:**

```
<device option_bit="bit number">
...
</device>
```

## Example




① ② ③ ④

**Block-by-block assignment:**

| | | |
|---|---|---|
| ① | DB9905.DBB0 | Device 1 |
| ② | DB9905.DBB4 | Device 2 |
| | ... | |
| ③ | DB9905.DBB192 | Device 49 |
| ④ | DB9905.DBB196 | Device 50 |

**Assignment to free locations:**

| | | |
|---|---|---|
| ① | DB9905.DBB0 | Device 1 |
| ② | DB9905.DBB4 | Device 2 |
| | ... | |
| ③ | DB9905.DBB72 | Device 19 |
| ④ | DB9905.DBB76 | Device 20 |
| | ... | |

## Feedback from the PLC

If dealers cannot integrate their own PLC logic, then the PLC feedback signals must be set by the script.

# 8.4 Display on the user interface

## Dialogs on the user interface

The following dialogs are available for Easy Extend:

- The control offers a **configurable dialog**, in which the available devices are shown.
- If first commissioning has not taken place yet, the control opens the **commissioning dialog**.

If a commissioning procedure is programmed for the unit (XML statement: "START_UP"), and the device has not been commissioned yet, the control starts the commissioning procedure.

This involves a complete data backup before the commissioning archives saved in the script file are imported. Standard or data class archives are permitted as archive types: *.arc and *.ard

- In the event of an error, the commissioning engineer can decide whether to roll back the commissioning procedure or to rectify possible errors in machine configurations manually.
- Commissioning can be aborted early with the "Cancel" function. The control then copies the previously saved commissioning files back.

If the machine has to be switched off after successful completion of the commissioning, the XML statement "POWER_OFF" can be used to program that a corresponding message is output on the control.

# 8.5 Generating language-dependent text

## Replacement characters for language-dependent text

The system offers the option of defining CONTROL properties (attribute values) during runtime. In order to use this function, the desired property must be set in a local variable and the variable name must be transferred to the tag as an attribute value preceded by $ characters.

**Example:**

```
<let name="caption_device_variable" type="string">my_device 3</let>
  <DEVICE option_bit="8">
    <!-- Direct: The entered expression is displayed. -->
    <NAME>my device 1</NAME>
  </DEVICE>
  <DEVICE option_bit="10">
    <!-- Indirect: The text saved for the text identifier caption_id_my_device2
is displayed. -->
    <NAME>$$caption_id_my_device2</NAME>
    <form>
      <init>
        <!-- Indirect: The text saved for the text identifier
form_caption_device2 is displayed. -->
        <caption>$$form_caption_device2</caption>
      </init>
    </form>
  </DEVICE>
  <!-- Indirect: The content of the variable caption_device_variable is dis-
played. -->
  <DEVICE option_bit="11">
    <NAME>$$$caption_device_variable</NAME>
  </DEVICE>
```

## Structure of text file

The XML files with the language-dependent texts must be created in UTF8 format:

**Example: oem_aggregate_eng.ts**

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TS>
<TS>
<context>
<name>EASY_EXTEND</name>
  <message>
    <source>form_caption_id_my_device2</source>
    <translation>my_device2</translation>
    <chars>64</chars>
  </message>
  <message>
    <source>form_caption_device2</source>
    <translation>my_input_form</translation>
    <chars>64</chars>
  </message>
</context>
</TS>
```

# 8.6 Examples

## 8.6.1 Example with parameters to support the commissioning

### Dialog with additional parameters

The input fields list selected drive parameters.

```
<DEVICE>
  <list_id>3</list_id>
  <name> "Test form" </name>
  <form>

  <init>
    <caption>Equipment Manager</caption>
    <control name = "edit1" xpos = "400" ypos = "34" refvar =
"drive/dc/p105[DO5]" />
    <control name = "edit1" xpos = "400" ypos = "54" refvar =
"$MC_AXCONF_MACHAX_USED[4]" />
    <control name = "edit1" xpos = "400" ypos = "74" refvar =
"drive/dc/p971[DO5]" />
    <control name = "edit1" xpos = "400" ypos = "94" refvar =
"drive/dc/r2[DO5]" />
  </init>

  <paint>
  <text xpos = "40" ypos = "34">dc[DO5]/p105</text>
  <text xpos = "40" ypos = "54">$MC_AXCONF_MACHAX_USED[4]</text>
  <text xpos = "40" ypos = "74">dc[DO5]/p971</text>
  <text xpos = "40" ypos = "94">dc[DO5]/r2</text>
  </paint>
  </form>

</DEVICE>
```

### Dialog with combo box

```
<form>

<init>
<caption>selected machine data</caption>
```

```
<DATA_ACCESS type="true" />
<!-- switch-on the direct access to the NC variables -->
<control name = "edit1" xpos = "322" ypos = "34"
refvar="$MN_AXCONF_MACHAX_NAME_TAB[0]" />
<control name = "edit2" xpos = "322" ypos = "54"
refvar="$MN_AXCONF_MACHAX_NAME_TAB[1]" />
<control name = "edit3" xpos = "322" ypos = "74"
refvar="$MN_AXCONF_MACHAX_NAME_TAB[2]" />
<control name = "edit4" xpos = "322" ypos = "94"
refvar="$MN_AXCONF_MACHAX_NAME_TAB[3]" />

<control name = "edit5" xpos = "322" ypos = "114" refvar="$MA_IS_ROT_AX[AX1]"
hotlink="true" />
<control name = "edit6" xpos = "322" ypos = "134" refvar="$MA_IS_ROT_AX[AX2]"
hotlink="true" />
<control name = "edit7" xpos = "322" ypos = "154" refvar="$MA_IS_ROT_AX[AX3]"
hotlink="true" />

<!-- using the control type combo box to display the rotation axis value -->
<control name = "edit5" xpos = "322" ypos = "194" refvar="$MA_IS_ROT_AX[AX1]"
fieldtype = "combobox" hotlink="true" >
<item value= "0" >no</item>
<item value= "1" >yes</item>
</control>

<control name = "edit6" xpos = "322" ypos = "214" refvar="$MA_IS_ROT_AX[AX2]"
fieldtype = "combobox" hotlink="true" >
<item value= "0" >No</item>
<item value= "1" >yes</item>
</control>

<control name = "edit7" xpos = "322" ypos = "234" refvar="$MA_IS_ROT_AX[AX3]"
fieldtype = "combobox" hotlink="true" >
<item value= "0" >No</item>
<item value= "1" >yes</item>
</control>

</init>

<paint>
<text xpos = "23" ypos = "34">AXCONF_MACHAX_TAB[0]</text>
<text xpos = "23" ypos = "54">AXCONF_MACHAX_TAB[1]</text>
<text xpos = "23" ypos = "74">AXCONF_MACHAX_TAB[2]</text>
<text xpos = "23" ypos = "94">AXCONF_MACHAX_TAB[3]</text>
<text xpos = "23" ypos = "114">Is rot axis 1</text>
<text xpos = "23" ypos = "134">Is rot axis 2</text>
```

```
<text xpos = "23" ypos = "154">Is rot axis 3</text>

<text xpos = "23" ypos = "174">using combo box control</text>

<text xpos = "23" ypos = "194">Is rot axis 1</text>
<text xpos = "23" ypos = "214">Is rot axis 2</text>
<text xpos = "23" ypos = "234">Is rot axis 3</text>

</paint>

</form>
```

## 8.6.2 Example with control elements

### Example of a combo box

If "combo box" is selected as the field type, the expressions to be displayed must also be defined. The <item> identifier should be used for this purpose. The combo box saves the index of the currently selected text in the variable belonging to the "CONTROL" (variable name). The index starts with 1.

```
<control name = "button1" xpos = "10" ypos = "10" fieldtype = "combobox">
<item>text1</item>
<item>text2</item>
<item>text3</item>
<item>text4</item>
</control>
```

### Example of a value assignment

If an arbitrary integer value is to be assigned to an expression, the attribute value="value" should be added to the identifier. Rather than consecutive numbers, the control variable now contains the item's assigned value.

```
<control name = "button1" xpos = "10" ypos = "10" fieldtype = "combobox">
<item value = "10" >text1</item>
<item value = "20" >text2</item>
<item value = "12" >text3</item>
<item value = "1" >text4</item>

</control>
```

**Note**

**"hotlink" attribute**

The hotlink attribute results in a cyclic update of the corresponding control. This means that when a value is entered, the following update cycle overwrites the entered value. To avoid this behavior, the immediate saving of the entries must be activated with the DATA_ACCESS tag.

Another possibility is to take the SOFTKEY_OK identifier into the form. This identifier is performed before the dialog is closed. In this block, the data comparison can be made between the control and reference variables with the UPDATE_CONTROLS statement.

## 8.6.3 Example of a power unit

**Note**

The two examples are also supplied on the SINUMERIK 828D Toolbox → Examples.

### Activating the drive object

The drive object to be activated has already been commissioned and deactivated again by the machine manufacturer to market the axis (axes) as an option.

To activate the axis, carry out the following steps:

- Activate the drive object via p0105.
- Enable 2nd axis in the channel machine data.
- Back up the drive machine data via p0971.
- Wait until the data has been written.
- Restart the NC and drives.

```
<DEVICE>
  <list_id>1</list_id>
  <name> "Activate the drive" </name>
  <SET_ACTIVE>
    <data name = "drive/dc/p105[DO5]">1</data>
    <data name = "$MC_AXCONF_MACHAX_USED[4]">5</data>
    <data name = "drive/dc/p971[DO5]">1</data>
    <while>
      <condition> "drive/dc/p971[DO5]" !=0 </condition>
    <while>
```

```
    <control_reset resetnc ="true" resetdrive = "true"/>
  </SET_ACTIVE>
  <SET_INACTIVE>
    <data name = "drive/dc/p105[DO5]">0</data>
    <data name = "$MC_AXCONF_MACHAX_USED[4]">0</data>
    <data name = "drive/dc/p971[DO5]">1</data>
    </while>
      <condition> "drive/dc/p971[DO5]" !=0 </condition>
    </while>
    <control_reset resetnc ="true" resetdrive = "true"/>
  </SET_INACTIVE>
</DEVICE>
```

## Activating the PLC-controlled device

The device is activated and deactivated by the user program using bit memory 100.0.

```
<AGM>
<OPTION_MD NAME= "$MN_USER_DATA_INT[0]" />
  <DEVICE>
    <LIST_ID>1</LIST_ID>
    <NAME> "PLC Device" </NAME>
    <SET_ACTIVE>
    <SET_ACTIVE>
      <DATA NAME= "PLC/M100.0"> 1 </DATA>
    </SET_ACTIVE>
    <SET_INACTIVE>
      <DATA NAME= "PLC/M100.0"> 0 </DATA>
    </SET_INACTIVE>
  </DEVICE>
</AGM>
```

# Tool management 9

## 9.1 Fundamentals

### 9.1.1 Tool Management - Overview

#### Tool management (TM)

The tool management (TM) function ensures that the right tool is in the right location on the machine at all times.

Machines, magazines, loading positions and tool buffers (e.g. spindles, grippers) form a specific system in which the tools are stored and transported. The tool management continually informs the NC of the current location of the tools and logs the tool movements initiated by the part program, PLC or SINUMERIK Operate.

During tool management commissioning the specific machine system architecture is mapped in the control. For example, one or more magazines are set up that are able to pick up tools at their locations. The "workplace" of a tool is described in the control in the form of value pairs (magazine number and location number).

---

**Note**

**Scope of delivery**

The tool management is included in the scope of delivery for all controls. The "Replacement tools for tool management" function is an option subject to license.

---

#### Replacement tools

If the "Replacement tools for tool management" option is used, replacement tools can be created and used.

Tools with the same name form a tool group and are distinguished by the duplo number. The maximum number of replacement tools of a tool group can be set via MD17500 $MN_MAXNUM_REPLACEMENT_TOOLS. The name of the tool group is programmed in the part program. Generally, the tool with the status "active" is used if the tool group has several tools. If this is blocked, one of the other tools from the same tool group is used.

### References

A comprehensive description of the tool and magazine parameters and the data structure can be found in:

- SINUMERIK 840D sl Description of Functions: Tool management

  As far as the range of functions is concerned, the NC part of this documentation is also valid for the SINUMERIK 828D. The description of the PLC functions and the communication between NC and PLC in this manual is **not** valid for the SINUMERIK 828D.
- SINUMERIK 828D List Manual: "Machine data"
- SINUMERIK 828D List Manual: "NC variable and interface signals"
- The setting of the tool management user interface is described in:
  SINUMERIK Operate for 828D Commissioning Manual

## 9.1.2 Structure of the tool management

### Function structure

The software components of the control have the following tasks in the tool management:

- **NC:**

  The tool management administrates the magazine locations. These locations might be empty, loaded with tools or assigned to oversized tools in adjacent locations. Empty locations can be loaded with other tools. The tool management provides the machine manufacturer with optimized management of tools and magazine locations. Magazine management provides extended functions such as load, unload or position tools. It also includes searches for tools, magazine locations and search strategies for replacement tools.

  For the tool monitoring functions, while the active monitoring is running, tools are disabled and no longer used. To continue machining, an equivalent tool (duplo tool) that is not blocked is used, if available.

  – TO area: All data that is required for the tool management (tool and magazine data) is in the NC in the TO area.

  – TO unit (TOA): For each channel, a maximum of one separate TO unit can be set up as subset of the NC TO area. A TO unit represents an independent area of the tool management. There is no connection between the TO units. Several channels can be assigned to one TO unit. However, one channel cannot be assigned to several TO units. Two-channel systems are preset, so that each channel is assigned its own TO unit (MD28085 $MC_MM_LINK_TOA_UNIT).

- **PLC:**
  - Execute tool change
  - Move tools in the magazine
  - Gripper control
  - Magazine control if applicable
  - Safety interlocks
  - Provides the structure of tool movements in transfer step tables
  - Acknowledge the tool movements with acknowledgment step tables
- **SINUMERIK Operate:**
  - Tool data display, input/output
  - Magazine data display, input/output
  - Load/unload relocation dialog

### PLC user program

The PLC user program executes the tool management jobs and acknowledges all position changes of the tools (and magazines). The monitoring and prevention of collisions is the task of the PLC user program alone, for example:

- Multiple spindles are using the same magazine.
- The paths of simultaneous jobs cross.
- As long as a large tool is located in the shifter, the chain must not be moved.

### PLC firmware

Functions of the PLC firmware:

- Assignment of tool management jobs to the PLC user program
- Communication of PLC user program acknowledgments to the tool management
- Transfer feedback signals for each acknowledgment (acknowledgment incorrect with error number or acknowledgment OK) to PLC user program.
- In addition: Register order status

## 9.1.3 Components of the tool management

### Tool list, magazines, magazine list

Circular and chain magazines can be managed. Other types of magazines are mapped on these. Loading points or loading stations shall be used as the magazine type for loading and unloading. A magazine buffer combines all other locations in which tools can be placed (spindle, gripper, ...).

**Default setting for tool magazines:**

As there must always be a buffer and a loading point, fewer real magazines can be managed. The number of magazines which the NCK can manage is specified by the system:

- PPU 24x.3/26x.3: Number of magazines = 3, of which 1 real magazine
- PPU 28x.3: Number of magazines = 4, of which 2 real magazines

  Special case for PPU 28x.3 (TE822): Number of magazines = 6, of which 1 real magazine per channel if channel 1 uses TO unit 1 and channel 2 uses TO unit 2 (default setting). If both channels use TO unit 1 jointly, up to 4 real magazines can be set up.

## Magazines

Information is provided by the system for all locations in the magazine, describing the content and status of the locations.

The position of a tool is described by an identifier for the magazine and an identifier for the location. Magazines have an identifier and a number, magazine locations only a number. In a real magazine (chain, circular, etc.), the position of the tool is identified by the magazine number assigned during start-up and the location within the magazine.

Example: The T number of the tool in magazine location 7 in magazine 1: $TC_MPP6[1,7]

## Tool list

The tool list contains all the tools known to the NC. These are the tools in the magazine and unloaded tools whose data is to be retained. The tool management works with loaded tools from the tool list.

## Magazine list

The magazine list is a location-oriented map of the tool magazine, gripper and spindle. The tool management only works with the tools from the magazine list. Additional tools without a magazine assignment can also be selected for tool changes. The tool must be inserted in the machine manually and removed again manually after machining (manual tool).

## Loading magazine

The loading magazine is the 1st internal magazine and is assigned magazine number 9999. The loading magazine has loading points for loading and unloading tools.

For the allocation of locations, one is fixed, all other locations can be assigned freely. Location 1 in the loading magazine is used for the fixed assignment. Location 1 is reserved for loading/unloading to all spindles/tool holders.

All positioning and relocation jobs to any locations (not loading points) are still handled via location 1. The stated jobs, which refer to a particular loading point, are output at the interface of this loading point. The loading points are assigned to magazines during start-up ($TC_MDP1). A loading point is an open access point to the magazine, where a tool is manually loaded and unloaded from the magazine.

## Buffer

Buffers are located in the 2nd internal magazine. The buffer includes the spindle, tool holder, gripper, loader and transfer location. The buffers are managed under magazine number 9998. Each buffer element is assigned a unique location. Any location numbers may be assigned. It is recommended that all spindles or tool holders be numbered in ascending order, starting at number 1. The assignment to real magazines or of spindles/tool holders to other buffers is made during start-up ($TC_MDP2, $TC_MLSR).

## Chain magazine

The setting in MD22550: $MC_TOOL_CHANGE_MODE may only have the value 1 for these magazine types.

Chain magazines do not as a rule have any additional buffer available for transportation between magazine and spindle. These additional buffers can contain tools temporarily.

Description of the buffers and loading points:

| Magazine | Location | Meaning |
|---|---|---|
| 1 | xx | Real magazine 1 (chain, plate, box), position xx |
| 9998 | 1 | Spindle |
| 9998 | 2 | Gripper |
| 9998 | 3 | Gripper |
| 9998 | 4 | Tool boy |
| 9998 | 5 | Shifter |
| 9999 | 1 | Loading point for spindle, manual tool |
| 9999 | 2 | Magazine loading point |

## Circular magazine

The setting of MD22550 $MC_TOOL_CHANGE_MODE is usually 0.

Circular magazines do not have any additional buffer with which tools can be transported from the magazine to the spindle. The tools on circular magazines are not physically transported into the spindle, but are moved into a defined position through rotation of the turret so that machining can take place with one particular tool. The tool is only transported to the spindle or tool holder in the software. Transporting the tool to the buffer 9998/1 (spindle) serves to inform the tool management that the turret holding the requested tool has been turned to the machining position.

The programming command T = identifier initiates the tool change. T = location can be programmed as an alternative. If T = location, no tool need actually be stored in the location.

Description of the buffers and loading points:

| Magazine | Location | Meaning |
|---|---|---|
| 1 | xx | Real magazine 1 (circular), position xx |
| 9998 | 1 | Tool holder |
| 9999 | 1 | Loading point for tool holder, manual tool |

If the value 1 is also set for the turret in MD22550: $MC_TOOL_CHANGE_MODE, the statement made for the chain magazines also applies here.

### Consider adjacent location

Consider adjacent location is used for oversized tools. When searching for empty locations (loading, tool change) the bits 4 ... 11 are then evaluated in the magazine location parameter $TC_MPP4 (half location occupied/reserved).

## 9.1.4 Multitool

### Multitool in the turning technology

A multitool is a mini turret with a number of tools that is loaded to a magazine and unloaded from a magazine as one tool in the turning technology. The T selection in the part program also detects and checks the tools of a multitool when searching for a tool.

Brief description:

- Using a multitool you have the possibility of storing more than one tool at a magazine location.
- A multitool has a defined number of multitool locations to accept tools.
- Regarding loading and unloading in a magazine, the multitool is treated as one tool.
- The multitool is considered to be an entity in its own right that is manually equipped, and which is loaded, unloaded or transported as a whole. The equipping of a multitool with tools or the removal of tools from the multitool do not create a tool management request to the PLC interface.
- The multitool knows the parameters "Number of locations" and "Type of clearance determination of the locations".
- The type of distance can be defined using the multitool location number or using the angle of the multitool location.
- Multitool names come from the same name space as the tool names and magazine names.
- Different multitools must have different names, i.e. a duplo number is not defined.

The functionality effects the following interfaces:

- NC programming language (data definition for the tool selection, tool change, tool movements)
- NC variables (data definition and visualization)
- NC/PLC interface

The description of the programming and the multitool data can be found in the following references:

### References

SINUMERIK 840D sl Function Manual Tool Management

## 9.1.5 Manual tools

### Manual tools

Manual tools are tools whose data is completely available in the NC, but are not loaded into the magazine. MD22562 $MC_TOOL_CHANGE_ERROR_MODE bit 1 is used to set as to whether manual tools are permissible in the system. The automatically selected tool must be inserted in the machine manually and removed again manually after machining.

The following types of tools are manual tools:

- Oversized tools
- Tools that cannot be stored in the magazine.
- Tools that may not be handled by the gripper system.

### Responsibility of the operator

The machine operator must ensure that:

- The data record of the tool positioned on the spindle is actually in the NC.
- The tool that corresponds to the data record in the NC is placed in the spindle.

| ⚠ CAUTION |
| --- |
| **Machine safety**<br>It is the responsibility of the machine operator to comply with the safety regulations via the PLC user program. |

### Unique identification of manual tools

A manual tool is allocated a unique identifier, $TC_TP8[T_No] bit 15 =1. If this state is set, a search is not made for an empty location in a magazine for the tool - and this (old) tool is directly transported to the virtual magazine location 9999/1 when the tool is changed. In the tool change subprogram, detecting the identifier can be performed with a bit-by-bit AND:

IF $TC_TP8[_TOOL_IN_SPINDLE] B_AND 'B1000000000000000' GOTOF HANDTOOL

**Setting the "manual tool" state:**

The "manual tool" state of a tool is set if the following conditions are fulfilled:

- The machine data is set: MD22562 $MC_TOOL_CHANGE_ERROR_MODE Bit 1 = 1
- As part of a tool change, an unloaded changing point was selected.

**Resetting the "manual tool" state:**

The "manual tool" state of a tool is reset in the following cases:

- The tool is unloaded, i.e. the tool no longer has a current magazine location. This can especially occur as a result of:
  - Unloading the old tool as part of the tool change.
  - Transporting a manual tool to a loading position (unloading as a result of an operator action at SINUMERIK Operate).
  - The command to select or load a manual tool was interrupted and the tool was still not moved to a magazine location.
- The tool was brought to a location in a real magazine or the loading magazine.

---

**Note**

**Alarms for manual tools**

An alarm (17212, 17214 or 17216) is always output to indicate that a tool change involving a manual tool has been executed. The alarm is reset by the tool change acknowledgment of the PLC user program.

The alarms can be hidden: MD11415 $MN_SUPPRESS_ALARM_MASK_2 Bit 16 = 1

**Loading a tool via SINUMERIK Operate**

A tool which was, for example, loaded to the spindle via SINUMERIK Operate, is not a manual tool, and as a consequence is placed with T0/M06 in the magazine.

---

# 9.2 Machine data for the tool management

## Machine data (default setting)

The following machine data has already been preset for tool management or will be set with "default data" during booting: These settings can be changed as necessary.

| Number | Name | Default |
|---|---|---|
| MD10715[0] | $MN_M_NO_FCT_CYCLE | 6 |
| MD10716[0] | $MN_M_NO_FCT_CYCLE_NAME | L6 |
| MD10717 | $MN_T_NO_FCT_CYCLE_NAME | TCHANGE |
| MD17500 | $MN_MAXNUM_REPLACEMENT_TOOLS | 1 |
| MD20124 | $MC_TOOL_MANAGEMENT_TOOLHOLDER | 1 |
| MD20310 | $MC_TOOL_MANAGEMENT_MASK | 181400F/180400F |
| MD22550 | $MC_TOOL_CHANGE_MODE | 0 / 1 |
| MD22560 | $MC_TOOL_CHANGE_MCODE | 206 |
| MD22562 | $MC_TOOL_CHANGE_ERROR_MODE | 0 (for manual tool) |

## MD20270: $MC_CUTTING_EDGE_DEFAULT

Basic setting of tool cutting edge without programming (DWORD): If no cutting edge is programmed after a tool change, the cutting edge number set in $MC_CUTTING_EDGE_DEFAULT is used.

| MD20270: $MC_CUTTING_EDGE_DEFAULT | |
|---|---|
| > 0 | Number of the cutting edge that is selected with M206 Cutting edge selection is also active if followed by D programming. |
| = 1 | Default setting |
| = 0 | No cutting edge is initially active after a tool change. Any tool offset active before tool change is deselected (corresponds to D0!). Cutting edge selection only active with D programming. |
| = -1 | Tool edge number of old tool also applies to new tool. |
| = -2 | Tool edge offset of old tool remains active until D is programmed. |

The setting in MD20270 affects the block preparation of the NC. To prevent preprocessing stop from occurring when the tool change command is called until it is acknowledged again, perform NC functions without tool offset in the tool change subprogram, for example, traversing axes or output of auxiliary functions.

**Example:** MD20270: $MC_CUTTING_EDGE_DEFAULT= 0 or = -2

After the tool change command M206, the axes can continue travel without having to wait for the tool change acknowledgment and execute traversing blocks without tool compensation. Travel only stops in a block with compensation selected (D no.) until end of tool change is signaled by the PLC.

**Part program:**

```
N10 T="Drill18"           ; Tool change preparation
N20 M6                    ; Tool change subprogram called
Tool change subprogram L6:
N10 M206                  ; Tool change
N20 D0                    : Compensation deselected
N40 Y150 M79              ; Traverse machine axes
N50 G01 D1 X10            ; Activate tool compensation.
                          ; Check whether tool has been changed.
                          ; Preprocessing stop is maintained until tool change
                          preparation is completed. The main run waits at N50 (D1)
                          until tool change has been executed and acknowledged.
```

## MD20310: $MC_TOOL_MANAGEMENT_MASK

Activating the tool management functions:

| MD20310: $MC_TOOL_MANAGEMENT_MASK | | |
|---|---|---|
| Bit 0 | = 1 | Tool management active:<br>The functions of tool management are enabled for the current channel. |
| Bit 1 | = 1 | Monitoring functions of tool management active:<br>The functions for monitoring tools (tool life and workpiece count) are enabled. |
| Bit 2 | = 1 | OEM functions active |
| Bit 3 | = 1 | Adjacent location consideration active |
| Bit 14 | = 1 | Tool and offset selection according to the settings in:<br>MD20110 $MC_RESET_MODE_MASK<br>MD20112 $MC_START_MODE_MASK |
| Bit 23 | = 1 | With offset selection no synchronization with main run. |
| Bit 24 | = 1 | Using an asynchronous transfer with acknowledgment status 201, you can move a tool to a location reserved for another tool with "reserved for tool from buffer".<br>This location reservation is then removed before the movement is executed ("Reserved for new tool to be loaded" (bit value="H8") remains effective). |
| Additionally only for T version: | | |
| Bit 16 | = 1 | T location number is active. |

If a locked tool is at the programmed location, the location of a replacement tool (if available) is output as job from the tool management when the "Replacement tools for tool management" option is set.

## MD52270: $MCS_TM_FUNCTION_MASK

Tool management function mask

| MD52270: $MCS_TM_FUNCTION_MASK | |
|---|---|
| Bit 0: | Creating a tool at the magazine location is not permitted.<br>Tools can only be created outside of the magazine. |
| Bit 1: | Load/unload lock, if the machine is not in the reset state.<br>Tools can only be loaded/unloaded when the relevant channel is in the reset state. |
| Bit 2: | Load/unload lock for EMERGENCY STOP.<br>Tools can only be loaded/unloaded when EMERGENCY STOP is not active. |
| Bit 3: | Tool in/out of spindle load/unload locked.<br>Tools cannot be loaded to or unloaded from the spindle. |
| Bit 4: | Loading is performed directly into the spindle.<br>Tools are only loaded directly into the spindle. |
| Bit 5: | Reserved |
| Bit 6: | Reserved |
| Bit 7: | Create tool using the T number.<br>The T number of the tool must be entered when creating a tool. |
| Bit 8: | Hide tool relocation.<br>The "Relocate tool" function is hidden in the user interface. |
| Bit 9: | Hide magazine positioning.<br>The "Position magazine" function is hidden in the user interface. |
| Bit 10: | Reactivate tool with magazine positioning.<br>Before reactivating, the tool is positioned at the loading point. |
| Bit 11: | Reactivate tool in all monitoring types.<br>When reactivating a tool, all the monitoring types for this tool enabled in the NC are reactivated. I.e. also the monitoring types that are not set for the relevant tool, but are only in the background. |
| Bit 12: | Hide reactivating tool.<br>The "Reactivate tool" function is hidden in the user interface. |

## Default setting for turning technology

The following machine data determines the function for turning technology:

- **MD22550: $MC_TOOL_CHANGE_MODE = 0**

  Setting for a revolver magazine: The new tool is changed immediately with the T function. No additional M command is used. No distinction is made between Prepare tool change and Execute tool change.

  The function "Manual tools" is not enabled for this case.

- **MD20310: $MC_TOOL_MANAGEMENT_MASK = 81400F (bit 16=1)**

  Bit 16 is used to set the tool programming type:

  T = "x" with x as tool identifier

  Tx, with x as location number of the magazine containing the tool used for machining.

  When the function is active, T1 selects the tool in location number 1 instead of the tool with identifier "1". Then the identifier of the tool in this location is ascertained (e.g. "FINISHING TOOL"). The procedure is as if T="FINISHING_TOOL" had been programmed.

  If T = location number, no tool need actually be stored in the location.

- **MD10717: $MN_T_NO_FCT_CYCLE_NAME = TCHANGE**

  Name of tool change cycle for T function replacement.

  For a description, refer to Section Example: Tool change cycle for turning machine (Page 312)

**ISO dialects in turning technology:**

Default setting for programming the tool offset:

MD10888 $MN_EXTERN_DIGITS_TOOL_NO = 0
MD10889 $MN_EXTERN_DIGITS_OFFSET_NO = 2
MD10890 $MN_EXTERN_TOOLPROG_MODE = 4

Programming T101 and T0101 leads to the same result: T1 H01

**References:** SINUMERIK 828D / 840D sl Function Manual: ISO dialects

- The description "Substitution by a replacement cycle for a tool selection block" is valid for the ISO mode (G291).
- The description "Tool change and tool offsets" is valid when programming the tool offset.

## Configuring the magazine

An NC POWER ON is needed after changing the magazine configuration. The changed configuration is only displayed after restarting the NC.

## See also

Configuration of a chain magazine with a dual gripper (Page 319)

Configuration of a circular magazine: (Page 306) The program is in the toolbox for SINUMERIK 828D in the "Examples" directory.

# 9.3 NC ↔ PLC user interface

## Overview

The tool management receives tasks for preparing and carrying out a tool change (T command, M06), a tool movement (MVTOOL) or magazine positioning (POSM) from the part program or from SINUMERIK Operate. From these tasks the TM defines the location change needed for the tool and assigns this to the PLC.

The user interface provides separate data blocks for loading, unloading, relocating and magazine positioning on the one hand and tool change on the other hand.

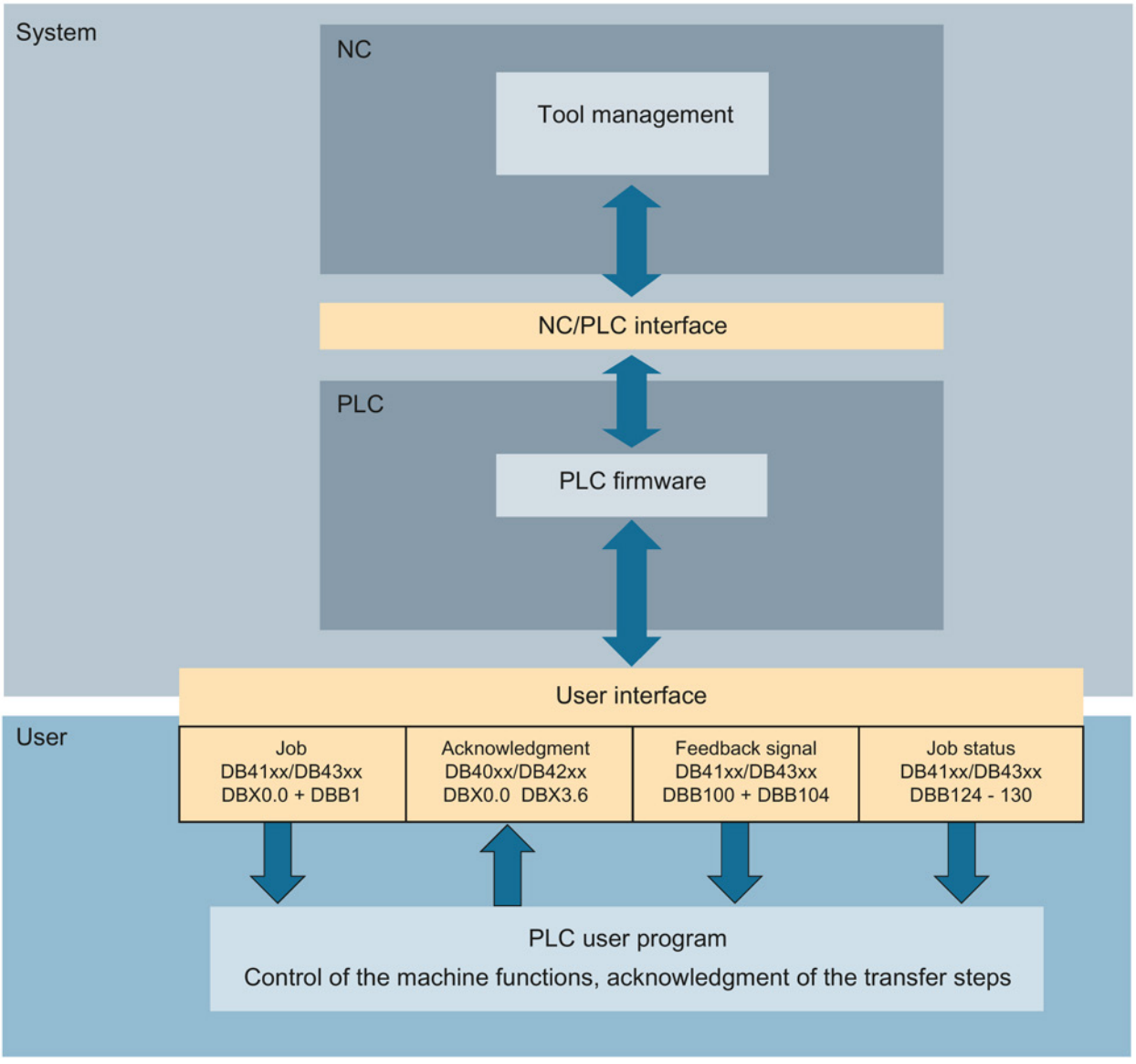



Figure 9-1 Interfaces of the tool management

## Loading/unloading and relocating

Loading/unloading and relocating: Tool movement (MVTOOL) or magazine positioning (POSM) from the part program or SINUMERIK Operate

xx: Loading station index

y : Index of the TO unit

- 40xx y000 Acknowledgments from the PLC user program for loading/unloading and relocating, magazine positioning
- 41xx y000 Jobs to the tool management for loading/unloading and relocating, magazine positioning
- 41xx y100 Feedback from the tool management
- 41xx y120 Job status

## Tool change

Tool change (T command, M06)

xx: Spindle index

y : Index of the TO unit

- 42xx y000 Prepare and execute acknowledgments from the PLC user program for the tool change
- 43xx y000 Prepare and execute jobs to the tool management for the tool change
- 43xx y100 Feedback from the tool management
- 43xx y120 Job status

Transfer and acknowledgment step tables:

- 9900 0000 Constant transfer step table (configurable)
- 9901 0000 Variable transfer step table (configurable and modifiable)
- 9902 0000 Acknowledgment step table (configurable)

**Example:**

There is a separate data area in the respective data blocks for each TO unit:

- Tool holder 1 in TO unit 1: DB4x00.0xxx
- Tool holder 2 in TO unit 1: DB4x01.0xxx
- Tool holder 1 in TO unit 2: DB4x00.1xxx
- Tool holder 2 in TO unit 2: DB4x01.1xxx

## 9.3.1 Relocating, unloading, loading tool, positioning magazine

### Tool and magazine movements

Per loading point there is one interface for:

- **Jobs**for loading, unloading and relocating a tool (MVTOOL) and for magazine positioning (POSM).

  Message to the PLC user program:

  – Job active

  – Job specification

  – Job description

  The jobs appear on the interface of the loading point from which the tool is to be exchanged.

- **Acknowledgments** of the PLC user program

  All acknowledgments for a job must take place in the interface of the same loading point. Acknowledgment errors should also be reset in this interface.

- **Feedback signal** from the tool management to the PLC user program

  Message to the PLC user program:

  – Status of acknowledgment

  – Error status

  – Map of the acknowledgment bits

- **Job status**

  Selected data from the last intermediate or end acknowledgment is saved. This data is needed by the PLC firmware for the next acknowledgment to the tool management and is readable for diagnostic purposes. This data can be used by the user program to restart after an abort (e.g. reset during a tool change).

All acknowledgments for a job must take place in the interface of the same loading point.

| Interface signal | Meaning |
|---|---|
| DB40xx.DBXy000.0 – DBXy003.6 | PLC user program:<br>Acknowledgments for loading/unloading/relocating or positioning the magazine |
| DB40xx.DBXy009.0 | PLC user program:<br>Resetting the message "Acknowledgment error" /DB41xx.DBX100.1) and the diagnostic information in the feedback interface |
| DB41xx.DBXy000.0 | Tool management:<br>Job for loading/unloading/relocating or positioning the magazine |
| DB41xx.DBBy001 | Tool management: Job specification |
| DB41xx.DBWy006 – DBWy034 | Job description |

| Interface signal | Meaning |
|---|---|
| DB41xx.DBXy100:0 | Positive feedback: Acknowledgment status, acknowledgment OK, one PLC cycle pending |
| DB41xx.DBXy100.1 | Negative feedback: Acknowledgment status, acknowledgment error, static pending |
| DB41xx.DBBy104 | Tool management: Feedback error status |
| DB41xx.DBXy108.0 - DB41xx.DBXy111.6 | Map of acknowledgments for load, unload, relocate or position magazine. This map belongs to the positive or negative feedback and remains valid for the same time. |
| DB41xx.DBWy124 – DBWy130 | Job status |

y [= 0 or 1]: TO unit 1 or TO unit 2

xx: Load location

## Jobs: Interface description

| DB4100...41xx | Signals from tool management [r] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| DBBy000 | Multitool: Data in the multitool area | | | | | | | Job is still active. |
| DBBy001 | | Multitool: Positioning is an NC activity | Multitool: Positioning | Positioning is an NC activity | Positioning | Relocating | Unloading | Loading |
| DBBy002 | Reserved | | | | | | | |
| DBBy003 | Reserved | | | | | | | |
| DBBy004 | Reserved | | | | | | | |
| DBBy005 | Reserved | | | | | | | |
| DBWy006 | Source magazine number (INT) | | | | | | | |
| DBWy008 | Source location number (INT) | | | | | | | |
| DBWy010 | Target magazine number (INT) | | | | | | | |
| DBWy012 | Target location number (INT) | | | | | | | |
| DBWy014: HMI → PLC | | | | | | | | Load/unload without moving magazine |
| DBWy016 | | | | | | | | |
| ... | | | | | | | | |
| DBWy056 | Multitool: Type (INT) | | | | | | | |
| DBWy058 | Multitool: Number of locations (INT) | | | | | | | |
| DBWy060 | Multitool: Tool distance (FLOAT) | | | | | | | |
| DBWy062 | | | | | | | | |
| DBWy064 | Multitool: Number (INT) | | | | | | | |

| DB4100...41xx | Signals from tool management [r] |
|---|---|
| DBWy066 | Multitool: Location number (INT) |
| DBWy068 | Multitool: Spindle number / tool holder number (INT) |

y [= 0 or 1]: TO unit 1 or TO unit 2

xx: Load location

## Jobs: Signal description

Signal description:

- Job:

  The interface contains a job. Job processing has not yet been completed with an end acknowledgment. This signal is reset after transmission of the end acknowledgment to the tool management.

- Multitool: Data in the multitool area

  The interface contains information about the multitool specified in the job.

- Loading:

  The "target location" parameter's magazine location should be loaded with a tool via the "source location" parameter's loading station.

- Unloading:

  The tool in the "source location" parameter's magazine location should be unloaded to the "target location" parameter's unloading station.

- Relocating:

  The tool in the "source location" parameter's magazine location should be relocated to the "target location" parameter's magazine location.

- Positioning:

  The "source location" parameter's magazine location should be positioned at the "target location" parameter's change/loading/unloading station. The tool remains in its magazine location.

- Positioning is an NC activity:

  The positioning job comes from the part program.

- Multitool: Positioning

  The tool in the multitool is moved to the machining position. The multitool remains in the tool holder / spindle.

- Multitool: Positioning is an NC activity:

  The positioning job comes from the part program.

- Source location:

  Magazine and location number of a tool that traverses or should be positioned at a change or loading station.

- Target location:

  Magazine and location number to where a tool is moving or to where a magazine location should be positioned.

- Loading/unloading without moving magazine:

  The signal is set by the HMI when reading in setup data or for the "Unload all tools" function in addition to the loading and unloading jobs. The signal can be used in the PLC user program to interlock undesirable magazine movements (e.g. set only total acknowledgment, no further action). The signal is reset by the system with a valid end acknowledgment for the job to the tool management.

- Multitool: Type:

  Type of distance coding: 1 = location number, 2 = length, 3 = angle

- Multitool: Number of locations

  Number of tool locations available in the multitool

- Multitool: Tool distance

  Value for the distance coding: Tool location number in the multitool, or angle or distance of the tool within the multitool

- Multitool: Number

  T number of the multitool

- Multitool: Location number

  Consecutively numbered location in the multitool

- Multitool: Spindle number / tool holder number

  Spindle number / tool holder number associated with the multitool

## Acknowledgments: Interface description

| DB4000...40xx | Signals to tool management [r/w] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| DBBy000 | Acknowl-edgment step 7 | Acknowl-edgment step 6 | Acknowl-edgment step 5 | Acknowl-edgment step 4 | Acknowl-edgment step 3 | Acknowl-edgment step 2 | Acknowl-edgment step 1 | Total ac-knowl-edgment |
| DBBy001 | Acknowl-edgment step 15 | Acknowl-edgment step 14 | Acknowl-edgment step 13 | Acknowl-edgment step 12 | Acknowl-edgment step 11 | Acknowl-edgment step 10 | Acknowl-edgment step 9 | Acknowl-edgment step 8 |
| DBBy002 | Acknowl-edgment step 23 | Acknowl-edgment step 22 | Acknowl-edgment step 21 | Acknowl-edgment step 20 | Acknowl-edgment step 19 | Acknowl-edgment step 18 | Acknowl-edgment step 17 | Acknowl-edgment step 16 |
| DBBy003 | Reserved | Acknowl-edgment step 30 | Acknowl-edgment step 29 | Acknowl-edgment step 28 | Acknowl-edgment step 27 | Acknowl-edgment step 26 | Acknowl-edgment step 25 | Acknowl-edgment step 24 |
| DBWy004 | Multitool: Location number (INT) | | | | | | | |
| DBBy006 | Reserved | | | | | | | |
| DBBy007 | Reserved | | | | | | | |

| DB4000...40xx | Signals to tool management [r/w] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DBBy008 | Reserved | | | | | | | |
| DBBy009 | | | | | | | | Resetting the ac-knowledg-ment error |
| DBBy010 | Reserved | | | | | | | |

y [= 0 or 1]: TO unit 1 or TO unit 2

xx: Load location

## Acknowledgments: Signal description

Signal description:

- Total acknowledgment:

  At a 0/1 edge, the end acknowledgment, with status 99, is sent to the current job (job complete, all target positions have been reached). As long as the signal is present, no changes may be made to the data of this interface!

  This signal is reset by the PLC firmware after the acknowledgment has been transferred to the tool management.

- Acknowledgment step 1…30:

  At a 0/1 edge, the appropriate acknowledgment step from the acknowledgment step table is sent to the tool management. As long as the signal is present, no changes may be made to the data of this interface and the variable transfer-step table!

  This signal is reset by the PLC firmware after the acknowledgment has been transferred to the tool management.

- Multitool: Location number

  The PLC user program returns the active location number to the NC tool management.

- Resetting the acknowledgment error:

  Resetting the message "Acknowledgment error" /DB41xx.DBX100.1) and the diagnostic information in the feedback interface.

## Feedback signals: Interface description

| DB4100...41xx | Signals from tool management [r] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| DBBy100 | | | | | | | Acknowl-edgment error | Acknowl-edgement OK |
| DBBy101 | Reserved | | | | | | | |
| DBWy102 | Reserved | | | | | | | |
| DBWy104 | Error status (WORD) | | | | | | | |
| DBBy106 | Reserved | | | | | | | |
| DBBy107 | Reserved | | | | | | | |
| DBBy108 | Acknowl-edgment step 7 | Acknowl-edgment step 6 | Acknowl-edgment step 5 | Acknowl-edgment step 4 | Acknowl-edgment step 3 | Acknowl-edgment step 2 | Acknowl-edgment step 1 | Acknowl-edgment step 0 |
| DBBy109 | Acknowl-edgment step 15 | Acknowl-edgment step 14 | Acknowl-edgment step 13 | Acknowl-edgment step 12 | Acknowl-edgment step 11 | Acknowl-edgment step 10 | Acknowl-edgment step 9 | Acknowl-edgment step 8 |
| DBBy110 | Acknowl-edgment step 22 | Acknowl-edgment step 22 | Acknowl-edgment step 21 | Acknowl-edgment step 20 | Acknowl-edgment step 19 | Acknowl-edgment step 18 | Acknowl-edgment step 17 | Acknowl-edgment step 16 |
| DBBy111 | Reserved | Acknowl-edgment step 30 | Acknowl-edgment step 29 | Acknowl-edgment step 28 | Acknowl-edgment step 27 | Acknowl-edgment step 26 | Acknowl-edgment step 25 | Acknowl-edgment step 24 |
| DBWy112 | Multitool: Location number (INT) | | | | | | | |

y [= 0 or 1]:with TO unit 1 or TO unit 2

xx: Load location

## Feedback signals: Signal description

Signal description:

- **Acknowledgment OK (DB41xx.DBXy100.0):**

  The acknowledgment of the PLC user program (area DB40xx.DBB0 to DBB3) was transferred without error to the tool management. This signal is reset after one PLC cycle.

- **Acknowledgment error (DB41xx.DBXy100.1):**

  Negative feedback of acknowledgment status. There is an error in the acknowledgment of the PLC user program (area DB40xx.DBB0 to DBB3). The cause of the error is displayed in "Error status".

  The "acknowledgment error" bit is only set if the acknowledgment from the PLC firmware is accepted as error free and the tool management detects and signals an error in the acknowledged tool transfer (for example, when the target location for tool transfer is occupied).

  Errors that are detected directly by the tool management in the NC before an acknowledgment is sent by the PLC user program, do not result in the setting of bit 100.1.

  If an error is present that prevents transfer of the acknowledgment to the tool management (error status 1 to 7), the error is only output on the interface of the tool management and not by the NC (no NC alarm!).

  If necessary, signal such errors with a user PLC alarm in the PLC user program.

  This signal remains pending statically until the error has been acknowledged (set the bit "Reset acknowledgment error" DB40xx.DBX9.0) by the user. The interface in DB40xx.DBB0 to DBB3 is disabled if bit "acknowledgment error" is pending. Incoming acknowledgment bits are not evaluated by the PLC firmware and are cleared when bit "Reset acknowledgment error" is set.

- **Multitool: Location number**

  Multitool: Location number that the PLC user program entered as active location number when acknowledging in the acknowledgment interface.

## Error status

When an error occurs, a diagnostics number not equal to zero is displayed in DB41xx.DBBy104 with the following meaning:

| Diagnostics number | Meaning |
|---|---|
| 0 | No error |
| 1 | Multiple acknowledgment signals at the same time |
| 2 | Acknowledgment without job |
| 3 | Invalid transfer step number |
| 4 | There is no job for a position specification |
| 5 | The status does not permit a location change (acknowledgment status 0 is used) |
| 7 | An impermissible acknowledgment status has been used |
| Other values: | The number corresponds to the error message of the tool management in the NC caused by this transfer. |

The error status is reset to zero by acknowledgment of the error by the user.

## Map of acknowledgments (DB41xx.DBBy108 to DBBy111)

The acknowledgments last set by the PLC user program (DB40xx.DBB0 to DBB3) are set and reset by the PLC firmware together with the bits "Acknowledgment OK" or "Acknowledgment error". If there is an error, the user uses these statically pending bits to see which acknowledgment step triggered the error. If the PLC user program incorrectly sets multiple acknowledgment bits, these are also entered one-to-one in the map.

## Job status

| DB4100...41xx | Signals from tool management [r] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBWy120 | Reserved | | | | | | | |
| DBWy122 | Reserved | | | | | | | |
| DBWy124 | Current magazine number of the tool (INT) | | | | | | | |
| DBWy126 | Current location number of the tool (INT) | | | | | | | |
| DBWy128 | Target magazine number of the tool (INT) | | | | | | | |
| DBWy130 | Target location number of the tool (INT) | | | | | | | |
| DBWy132 | Reserved | | | | | | | |
| DBWy134 | Reserved | | | | | | | |
| DBWy136 | Reserved | | | | | | | |
| DBWy138 | Reserved | | | | | | | |
| DBWy140 | Multitool type (INT) | | | | | | | |
| DBWy142 | Multitool number of locations (INT) | | | | | | | |
| DBWy144 | Multitool tool distance (FLOAT) | | | | | | | |
| DBWy146 | | | | | | | | |
| DBWy148 | Multitool location number (INT) | | | | | | | |

y [= 0 or 1]: TO unit 1 or TO unit 2

xx: Load location

## 9.3.2 Tool change

### Interface description

Per tool holder/spindle there is one interface for:

- Jobs for preparing and executing the tool change.

  Message to the PLC user program: Job active; job specification and job description.

  The jobs appear in the tool holder interface (of the spindle) where a tool is to be exchanged.

- Acknowledgments of the PLC user program

  All acknowledgments for a job must take place in the interface of the same tool holder (of the same spindle). Acknowledgment errors should also be reset in this interface.

- Feedback signal from the tool management to the PLC user program

  Message to the PLC user program: Acknowledgment status, error status, map of the acknowledgment bits.

- Job status

  Selected data from the last intermediate or end acknowledgment is saved. This data is needed by the PLC firmware for the next acknowledgment to the tool management and is readable for diagnostic purposes. This data can be used by the user program to restart after an abort (e.g. reset during a tool change).

### End acknowledgment for tool change

A common end acknowledgment (tool change via T command using turret) or separate end acknowledgment (Txx and M206 in separate blocks, default milling) is possible for "Prepare tool change" and "Implement tool change". With the appropriate MD setting, the end acknowledgment for the job to prepare for a tool change permits the NC preprocessing to continue.

The machine data with which the response of block preprocessing, main run, and the various acknowledgment responses are defined are described in Chapter Machine data for the tool management (Page 257).

The main NC run can be continued with the end acknowledgment to the job "Carry out tool change". Therefore, these end acknowledgments should take place as early as possible. This can mean that the end acknowledgment can take place before the old tool is in the magazine (e.g. the new tool is in the spindle, the old tool in the tool boy). The rest of the steps to bring the old tool into the magazine must then be communicated asynchronously. The same interface should be used as for synchronous acknowledgments.

| Interface signal | Meaning |
|---|---|
| DB42xx.DBXy000.0 – DBXy003.6 | PLC user program:<br>Preparing and carrying out acknowledgments for tool change |
| DB42xx.DBXy009.0 | PLC user program:<br>Resetting the message "Acknowledgment error" (DB43xx.DBX100.1) and the diagnostic information in the feedback interface |
| DB43xx.DBXy000.0 | Tool management: Job for "Prepare tool change" and "Execute tool change" |
| DB43xx.DBBy001 | Tool management job specification |
| DB43xx.DBWy006 – DBWy034 | Job description |
| DB43xx.DBXy100.0<br>DB43xx.DBXy100.1 | Positive feedback: Acknowledgment status, acknowledgment OK, one PLC cycle pending<br>Negative feedback: Acknowledgment status, acknowledgment error, static pending |
| DB43xx.DBXy100.0<br>DB43xx.DBXy100.1 | Positive feedback: Acknowledgment status, one PLC cycle pending<br>Negative feedback: Acknowledgment status, static pending |
| DB43xx.DBXy100.0 | Tool management feedback: Acknowledgment status |
| DB43xx.DBBy104 | Tool management feedback: Error status |
| DB43xx.DBXy108.0 - DB43xx.DBXy111.6 | Map of acknowledgments for tool change: This map belongs to the positive or negative feedback and remains valid for the same time |
| DB43xx.DBWy124 – DBWy138 | Job status |

y [= 0 or 1]: TO unit 1 or TO unit 2

xx: Spindle index / tool holder

## Jobs: Interface description

| DB4300...43xx | Signals from tool management [r] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBBy000 | Multitool: Data in the multi-tool area | -- | -- | -- | -- | -- | -- | Job is still active. |
| DBBy001 | Tool remains in the spindle | Unload manual tool | Load manual tool | No old tool | T0 | Prepare change | Change tool (initiated by: M06) | Fixed-location coded |
| DBBy002 | Reserved | | | | | | | |
| DBBy003 | Reserved | | | | | | | |

| DB4300...43xx | Signals from tool management [r] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DBBy004 | Reserved | | | | | | | |
| DBBy005 | Reserved | | | | | | | |
| DBWy006 | Source magazine number for new tool (INT) | | | | | | | |
| DBWy008 | Source location number for new tool (INT) | | | | | | | |
| DBWy010 | Reserved | | | | | | | |
| DBWy012 | Reserved | | | | | | | |
| DBWy014 | Reserved | | | | | | | |
| DBWy016 | Reserved | | | | | | | |
| DBWy018 | Target magazine number for old tool (INT) | | | | | | | |
| DBWy020 | Target location number for old tool (INT) | | | | | | | |
| DBWy022 | Location type (INT) | | | | | | | |
| DBWy024 | Size left (INT) | | | | | | | |
| DBWy026 | Size right (INT) | | | | | | | |
| DBWy028 | Reserved | | | | | | | |
| DBWy030 | Reserved | | | | | | | |
| DBBy032 | Tool status for a new tool: | | | | | | | |
| | Tool has been in use | Tool fixed-location coded | Tool being changed | Prewarning limit reached | Measuring tools | Tool disabled | Tool enabled | Active tool |
| DBBy033 | Tool status for a new tool: | | | | | | | |
| | Manual tool | 1:1 Change | -- | Master tool | To be loaded | To be unloaded | Blocked | Identification for tools in the buffer |
| DBWy034 | New tool: Internal T number of the NC (INT) | | | | | | | |
| DBWy036 | Reserved | | | | | | | |
| DBWy038 | Reserved | | | | | | | |
| DBWy040 | Reserved | | | | | | | |
| DBWy042 | Reserved | | | | | | | |
| DBWy044 | Free parameter 1 (DWORD) | | | | | | | |
| DBWy046 | | | | | | | | |
| DBWy048 | Free parameter 2 (DWORD) | | | | | | | |
| DBWy050 | | | | | | | | |
| DBWy052 | Free parameter 3 (DWORD) | | | | | | | |
| DBWy054 | | | | | | | | |
| DBWy056 | Multitool: Type (INT) | | | | | | | |
| DBWy058 | Multitool: Number of locations (INT) | | | | | | | |
| DBWy060 | Multitool: Tool distance (FLOAT) | | | | | | | |
| DBWy062 | | | | | | | | |
| DBWy064 | Multitool: Number (source) of the new tool (INT) | | | | | | | |
| DBWy066 | Multitool: Location number (source) of the new tool (INT) | | | | | | | |
| DBWy068 | Multitool: Number (target) of the old tool (INT) | | | | | | | |
| DBWy070 | Multitool: Location number (target) of the old tool (INT) | | | | | | | |

| DB4300...43xx | Signals from tool management [r] |
|---|---|
| DBWy072 | Location type (INT) |
| DBWy074 | Size left (INT) |
| DBWy076 | Size right (INT) |
| DBWy078 | Size top (INT) |
| DBWy080 | Size bottom (INT) |
| DBWy082 | Tool status for new tool (INT) |
| DBWy084 | T number of the tool (INT) |
| DBWy086 | Spindle number / tool holder number (INT) |
| DBWy088 | Multitool: Original magazine number (INT) |
| DBWy090 | Multitool: Original location number (INT) |

y [= 0 or 1]: TO unit 1 or TO unit 2

xx: Tool holder

## Jobs: Signal description

Signal description:

- Job:

  The interface contains a job. Job processing has not yet been completed with an end acknowledgment. This signal is reset after transmission of the end acknowledgment to the tool management.

- Multitool: Data in the multitool area

  The interface contains information about the multitool specified in the job.

- Fixed-location coded: The new tool is fixed-location coded.

- Execute tool change:

  The new tool is to be loaded into the tool holder / the spindle. The old tool is to be brought back to a magazine location. This job always requires an end acknowledgment.

- Prepare a tool change:

  Initialize new tools. If necessary, position magazine location for old tool at the changing point. This job requires an individual end acknowledgment. If there is a parallel job "Execute change," end acknowledgment for the preparation is not necessary.

- T0: T0 has been programmed (empty tool holder / spindle).

- No old tool:

  Tool change into the previously empty tool holder / spindle.

- Load manual tool:

  A manual tool is to be loaded. SINUMERIK Operate displays the tool which is to be loaded.

- Unload manual tool:

  The tool is to be changed via manual operation.

- Tool remains in spindle:

  The bit is set at a change from tool holder → spindle to tool holder → spindle. Triggers can be, for example, reset start mode or block search.

- Source location for the new tool:

  Magazine and location number from where the new tool comes (mostly a location in a real magazine).

- Target location for the old tool:

  Magazine and location number to where the old tool is to be transported (mostly a location in a real magazine).

- Origin of the new tool:
  - Internal T number: Internal T number of the new tool
  - Tool status: Tool status of the new tool
  - Location type: Location type of the new tool
  - Size: Size (right, left) of the new tool
- User-definable parameters: Three user-definable parameters which are transferred by the part program to the PLC user program.
- Multitool: Type:

  Type of distance coding: 1 = location number, 2 = length, 3 = angle

- Multitool: Number of locations

  Number of tool locations available in the multitool

- Multitool: Tool distance

  Value for the distance coding: Tool location number in the multitool, or angle or distance of the tool within the multitool

- Multitool: Number (source) / multitool: Location number (source) of the new tool

  Magazine number / location number from where the new multitool comes (mostly a location in a real magazine).

- Multitool: Number (target) / multitool: Location number (target) of the old tool

  Magazine number / location number to where the old multitool is to be transported (mostly a location in a real magazine).

- Location type: Location type of the new tool
- Size left: Space required for the multitool in the magazine at the left
- Size right: Space required for the multitool in the magazine at the right
- Size top: Space required for the multitool in the magazine at the top
- Size bottom: Space required for the multitool in the magazine at the top
- Tool status
- T number of the tool
- Spindle number / tool holder number

- Multitool: Original magazine number
- Multitool: Original location number

## Acknowledgments: Interface description

| DB4200 ... 42xx | Signals to tool management [r/w] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| DBBy000 | Acknowl-edgment step 7 | Acknowl-edgment step 6 | Acknowl-edgment step 5 | Acknowl-edgment step 4 | Acknowl-edgment step 3 | Acknowl-edgment step 2 | Acknowl-edgment step 1 | Total ac-knowl-edgment |
| DBBy001 | Acknowl-edgment step 15 | Acknowl-edgment step 14 | Acknowl-edgment step 13 | Acknowl-edgment step 12 | Acknowl-edgment step 11 | Acknowl-edgment step 10 | Acknowl-edgment step 9 | Acknowl-edgment step 8 |
| DBBy002 | Acknowl-edgment step 23 | Acknowl-edgment step 22 | Acknowl-edgment step 21 | Acknowl-edgment step 20 | Acknowl-edgment step 19 | Acknowl-edgment step 18 | Acknowl-edgment step 17 | Acknowl-edgment step 16 |
| DBBy003 | Reserved | Acknowl-edgment step 30 | Acknowl-edgment step 29 | Acknowl-edgment step 28 | Acknowl-edgment step 27 | Acknowl-edgment step 26 | Acknowl-edgment step 25 | Acknowl-edgment step 24 |
| DBWy004 | Multitool: Location number (INT) | | | | | | | |
| DBBy006 | Reserved | | | | | | | |
| DBBy007 | Reserved | | | | | | | |
| DBBy008 | Reserved | | | | | | | |
| DBBy009 | | | | | | | Status 3 for total acknowl-edgment | Resetting the acknowl-edgment error |
| DBBy10 | Reserved | | | | | | | |

y [= 0 or 1]: TO unit 1 or TO unit 2

xx: Tool holder

## Acknowledgments: Signal description

Signal description:

- Total acknowledgment:

  At a 0/1 edge, the end acknowledgment, with status 99, is sent to the current job (job complete, all target positions have been reached). As long as the signal is present, no changes may be made to the data of this interface!

  This signal is reset by the PLC firmware after the acknowledgment has been transferred to the tool management.

- Acknowledgment step 1…30:

  At a 0/1 edge, the appropriate acknowledgment step from the acknowledgment step table is sent to the tool management. As long as the signal is present, no changes may be made to the data of this interface and the variable transfer-step table!

  This signal is reset by the PLC firmware after the acknowledgment has been transferred to the tool management.

- Multitool: Location number

  The PLC user program returns the active location number to the NC tool management.

- Resetting the acknowledgment error:

  Resetting the message Acknowledgment error (DB43xx.DBX100.1) and the diagnostic information in the feedback interface.

- Status 3 for total acknowledgment

  Bit to cancel the change job:

  – The use is the same as with a configured acknowledgment step, status=3.

  – The use is exclusively the responsibility of the machine builder in the PLC user program.

## Feedback signals: Interface description

| DB4300...43xx | Signals from tool management [r] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBBy100 | | | | | | | Acknowl-edgment error | Acknowl-edgement OK |
| DBBy101 | Reserved | | | | | | | |
| DBWy102 | Reserved | | | | | | | |
| DBWy104 | Error status (WORD) | | | | | | | |
| DBWy106 | Reserved | | | | | | | |
| DBBy108 | Acknowl-edgment step 7 | Acknowl-edgment step 6 | Acknowl-edgment step 5 | Acknowl-edgment step 4 | Acknowl-edgment step 3 | Acknowl-edgment step 2 | Acknowl-edgment step 1 | Acknowl-edgment step 0 |
| DBBy109 | Acknowl-edgment step 15 | Acknowl-edgment step 14 | Acknowl-edgment step 13 | Acknowl-edgment step 12 | Acknowl-edgment step 11 | Acknowl-edgment step 10 | Acknowl-edgment step 9 | Acknowl-edgment step 8 |

| DB4300...43xx | Signals from tool management [r] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DBBy110 | Acknowl-edgment step 23 | Acknowl-edgment step 22 | Acknowl-edgment step 21 | Acknowl-edgment step 20 | Acknowl-edgment step 19 | Acknowl-edgment step 18 | Acknowl-edgment step 17 | Acknowl-edgment step 16 |
| DBBy111 | Reserved | Acknowl-edgment step 30 | Acknowl-edgment step 29 | Acknowl-edgment step 28 | Acknowl-edgment step 27 | Acknowl-edgment step 26 | Acknowl-edgment step 25 | Acknowl-edgment step 24 |
| DBBy112 | Multitool: Location number (INT) | | | | | | | |

y [= 0 or 1]: TO unit 1 or TO unit 2

xx: Tool holder

## Feedback signals: Signal description

Signal description:

- **Acknowledgment OK (DB43xx.DBX100.0): Positive feedback of acknowledgment status.**

  The acknowledgment of the PLC user program (area DB42xx.DBB0 to DBB3) was transferred without error to the tool management. This signal is reset after one PLC cycle.

- **Acknowledgment error (DB43xx.DBX100.1): Negative feedback of acknowledgment status.**

  There is an error in the acknowledgment of the PLC user program (area DB42xx.DBB0 to DBB3). The cause of the error is displayed in "Error status".

  The "acknowledgment error" bit is only set if the acknowledgment from the PLC firmware is accepted as error free and the tool management detects and signals an error in the acknowledged tool transfer (for example, when the target location for tool transfer is occupied).

  Errors that are detected directly by the tool management in the NC before an acknowledgment is sent by the PLC user program, do not result in the setting of bit 100.1.

  If an error is present that prevents transfer of the acknowledgment to the tool management (error status 1 to 7), the error is only output on the interface of the tool management and not by the NC (no NC alarm!).

  If necessary, signal such errors with a user PLC alarm in the PLC user program.

  This signal remains pending statically until the error has been acknowledged (set the bit "Reset acknowledgment error" DB4200.DBX9.0) by the user. The interface in DB42xx.DBB0 to DBB3 is disabled if bit "acknowledgment error" is pending. Incoming acknowledgment bits are not evaluated by the PLC firmware and are also cleared when bit "Reset acknowledgment error" is set.

- **Multitool: Location number**

  Multitool: Location number that the PLC user program entered as active location number when acknowledging in the acknowledgment interface.

## Error status

When an error occurs, a diagnostics number not equal to zero is displayed in DB43xx.DBB104 with the following meaning:

| Diagnostics number | Meaning |
|---|---|
| 0 | No error |
| 1 | Multiple acknowledgment signals at the same time |
| 2 | Acknowledgment without job |
| 3 | Invalid transfer step number |
| 4 | There is no job for a position specification |
| 5 | The status does not permit a location change (acknowledgment status 0 is used) |
| 7 | An impermissible acknowledgment status has been used |
| Other values: | The number corresponds to the error message of the tool management in the NC caused by this transfer. |

The error status is reset by acknowledgment of the error by the user.

## Map of acknowledgments (DB43xx.DBB108 to DBB111)

The acknowledgments last set by the PLC user program (DB42xx.DBB0 to DBB3) are set and reset by the PLC firmware together with the bits "Acknowledgment OK" or "Acknowledgment error." If there is an error, the user uses these statically pending bits to see which acknowledgment step triggered the error. If the PLC user program incorrectly set multiple acknowledgment bits, these are also entered one-to-one in the map.

## Job status

| DB4300 ... 43xx | Signals from tool management [r] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBWy120 | Reserved | | | | | | | |
| DBWy122 | Reserved | | | | | | | |
| DBWy124 | Current magazine number for new tool (INT) | | | | | | | |
| DBWy126 | Current location number for new tool (INT) | | | | | | | |
| DBWy128 | Target magazine number for new tool (INT) | | | | | | | |
| DBWy130 | Target location number for new tool (INT) | | | | | | | |
| DBWy132 | Current magazine number for old tool (INT) | | | | | | | |
| DBWy134 | Current location number for old tool (INT) | | | | | | | |
| DBWy136 | Target magazine number for old tool (INT) | | | | | | | |
| DBWy138 | Target location number for old tool (INT) | | | | | | | |
| DBWy140 | Multitool: Type (INT) | | | | | | | |
| DBWy142 | Multitool: Number of locations (INT) | | | | | | | |

| DB4300 ... 43xx | Signals from tool management [r] |
|---|---|
| DBWy144 | Multitool: Tool distance (FLOAT) |
| DBWy146 | |
| DBWy148 | Multitool: Location number (INT) |

y [= 0 or 1]: TO unit 1 or TO unit 2

xx: Tool holder

## 9.3.3 Transfer-step and acknowledgment-step tables

### Configurable step tables

There are configurable tables available in the data blocks TM_CTS (DB9900), TM_VTS (DB9901) and TM_ACK (DB9902), which are used to describe the tool movement sequences.

| Interface signal | Name | Meaning |
|---|---|---|
| DB9900 | TM_CTS | Constant transfer-step table (configurable) |
| DB9901 | TM_VTS | Variable transfer-step table (can be configured and written to from the PLC user program) |
| DB9902 | TM_ACK | Acknowledgment-step table (configurable) |

The data blocks DB40xx, 41xx, 42xx and 43xx are system blocks and are created automatically by the control.

The data blocks DB9900, DB9901 and DB9902 are made available by the Programming Tool under Libraries / Special data blocks. The blocks are not yet filled with the necessary data. The user must copy them into the PLC project and edit them.

### Transfer-step tables

The individual tool movements are defined as transfer steps – tool from magazine location x/y to magazine location m/n. Acknowledgment steps can be defined with these transfer steps. DB9900 contains permanently configured transfer steps (constant transfer-step table). DB9901 can be changed by the PLC user program; for example, for acknowledging intermediate steps such as magazine locations for tool change preparation (variable transfer step table).

| DB9900 | Constant transfer-step table [r] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBW0 | Transfer step 1 Source magazine number (INT) | | | | | | | |
| DBW2 | Transfer step 1 Source location number (INT) | | | | | | | |
| DBW4 | Transfer step 1 Target magazine number (INT) | | | | | | | |
| DBW6 | Transfer step 1 Target location number (INT) | | | | | | | |
| DBW8 | Transfer step 2 Source magazine number (INT) | | | | | | | |
| DBW10 | Transfer step 2 Source location number (INT) | | | | | | | |

| DB9900 | Constant transfer-step table [r] |
|---|---|
| DBW12 | Transfer step 2 Target magazine number (INT) |
| DBW14 | Transfer step 2 Target location number (INT) |
| | … |
| DBW504 | Transfer step 64 Source magazine number (INT) |
| DBW506 | Transfer step 64 Source location number (INT) |
| DBW508 | Transfer step 64 Target magazine number (INT) |
| DBW510 | Transfer step 64 Target location number (INT) |

| DB9901 | Variable transfer-step table [rw] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBW0 | Transfer step 101 Source magazine number (INT) | | | | | | | |
| DBW2 | Transfer step 101 Source location number (INT) | | | | | | | |
| DBW4 | Transfer step 101 Target magazine number (INT) | | | | | | | |
| DBW6 | Transfer step 101 Target location number (INT) | | | | | | | |
| DBB 8 | Transfer step 102 Source magazine number (INT) | | | | | | | |
| DBW10 | Transfer step 102 Source location number (INT) | | | | | | | |
| DBW12 | Transfer step 102 Target magazine number (INT) | | | | | | | |
| DBW14 | Transfer step 102 Target location number (INT) | | | | | | | |
| | … | | | | | | | |
| DBW504 | Transfer step 164 Source magazine number (INT) | | | | | | | |
| DBW506 | Transfer step 164 Source location number (INT) | | | | | | | |
| DBW508 | Transfer step 164 Target magazine number (INT) | | | | | | | |
| DBW510 | Transfer step 164 Target location number (INT) | | | | | | | |

## Acknowledgment step table

Each entry indexes two transfer steps (for the new and old tool) and gives the corresponding status reached. The acknowledgment-step table in DB9902 is used jointly for acknowledgments on the interface of the loading point and on the interface of the tool holder.

| DB9902 | Acknowledgment-step table [r] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBB0 | Acknowledgment step 1 Transfer step for new tool (BYTE) | | | | | | | |
| DBB1 | Acknowledgment step 1 Transfer step for old tool (BYTE) | | | | | | | |
| DBB2 | Acknowledgment step 1 Acknowledgment status (BYTE) | | | | | | | |
| DBB3 | Acknowledgment step 1 reserved | | | | | | | |
| DBB4 | Acknowledgment step 2 Transfer step for new tool (BYTE) | | | | | | | |
| DBB5 | Acknowledgment step 2 Transfer step for old tool (BYTE) | | | | | | | |
| DBB6 | Acknowledgment step 2 Acknowledgment status (BYTE) | | | | | | | |
| DBB7 | Acknowledgment step 2 reserved | | | | | | | |
| | … | | | | | | | |
| DBB116 | Acknowledgment step 30 Transfer step for new tool (BYTE) | | | | | | | |

| **DB9902** | **Acknowledgment-step table [r]** |
|---|---|
| DBB117 | Acknowledgment step 30 Transfer step for old tool (BYTE) |
| DBB118 | Acknowledgment step 30 Acknowledgment status (BYTE) |
| DBB119 | Acknowledgment step 30 reserved |

## See also

PLC Program Blocks (Page 284)

# 9.4 PLC Program Blocks

## 9.4.1 Acknowledgment process

### Information to tool management

The tool management expects acknowledgment of its orders in order to track and carry the actual tool positions. At least one acknowledgment is required for each order. This is sufficient for many applications.

Acknowledgment takes place either via the table defined in DB9902, or in one step with the total acknowledgment (DB40xx/42xx DBX0.0) after a completely finished tool management order using a 0/1 edge (set) of the corresponding bits in the user interface.

As long as the acknowledgment signal is present, no changes may be made to the data of this interface! This signal is reset by the PLC firmware after the acknowledgment has been transmitted to the tool management. In certain circumstances resetting may take place after several PLC cycles.

There are further advantages if the tool management is also informed of intermediate positions:

- Information about intermediate positions:

  If the intermediate tool positions are known in the tool management, the allocation of the buffer magazine may be queried. This makes powering up easier after switching off and on again or after canceling a command (e.g. via reset). If the tool currently being changed is needed again immediately, it can be loaded back into the spindle from a buffer location without loading it into the magazine first.

- Information about magazine positions:

  If the tool management knows at which magazine location there is a transfer point (changing point for the spindle, loading point), it can determine the shortest paths in the magazine for finding an empty location or for selecting a new tool. During orders the tool management can usually recognize the magazine position from intermediate acknowledgments (e.g. tool transfer between the real magazine and the buffer) or from the end acknowledgment (e.g. "magazine positioning" order complete). If the magazine is positioned by the PLC user program itself (e.g. by means of HMI or machine keys) without a tool management order, this must be communicated to the tool management via synchronous signals.

## 9.4.2 Types of acknowledgment

### Tool and magazine movements

The tool management distinguishes between synchronous acknowledgment and an asynchronous job-independent message.

**Synchronous acknowledgment**

- Acknowledgment of the intermediate steps of a job (tool management registers the current position changes of tools, part program must wait).

  The tool management is informed of intermediate steps of a job by an intermediate acknowledgment. In intermediate acknowledgments, only the target position of the intermediate step is relevant. The source position is known from the job or the last intermediate acknowledgment. During a tool change, two tools (new and old) can also be acknowledged simultaneously. Intermediate acknowledgments are only possible **before** the end acknowledgment.

- End acknowledgment of a job (part program can be continued)

  An end acknowledgment is necessary for every job. The end acknowledgment allows the part program to continue and frees up the job interface for new jobs. It should take place as soon as possible (e.g. as soon as the new tool is in the spindle and a collision is no longer expected). The tool management can be informed asynchronously of further steps after the end acknowledgment of a job (e.g. the return path of the old tool into the magazine).

**Asynchronous job-independent message**

Position change of a tool or magazine ("asynchronous message", e.g. when the PLC changes a tool position via machine control panel operation without tool change job).

An asynchronous message can be used to inform the tool management of a tool or magazine movement, independently of a job. An asynchronous message must always contain a source position (from) **and** target position (to).

Tool movements within a magazine (relocate tool) can only be carried out with magazine locations which are actually occupied. Empty transfers are not permissible. Two asynchronous transfers can be implemented in a single message. In this case the interface must be used for tool change DB42xx.

### Effect of acknowledgments

Effect of acknowledgments on the job and the part program:

- Intermediate and end acknowledgment take place synchronously with the job.
  - The part program must wait.
  - No new job can come yet.
- Message of an asynchronous transfer:
  - The part program continues to run.
  - The message is completely independent of any job.

## 9.4.3 Acknowledgment states

### Acknowledgment states and their meaning:

The status indicated by the respective type of acknowledgment is shown in the following table:

| Acknowledgment | | Meaning |
|---|---|---|
| Synchronous end acknowledgment | 1 | Job finished at the specified position:<br>The tools are in the positions specified. The part program can be continued. |
| | 3 | Job canceled:<br>The job is canceled, previously acknowledged tool position changes will be kept. The cancel command itself does not trigger any position acknowledgments or changes in the tool management. |
| | 7 | Repeat "prepare tool change" job:<br>The tool change was informed of a new tool position in advance. The "prepare tool change" job is to be recalculated with this position. This is permissible only for preparation commands which have not yet been acknowledged. |
| | 99 | Total acknowledgment: Job complete, all positions reached.<br>All the tools concerned are in the positions specified in the job. The part program can be continued. All target positions from the job have been reached. |
| Synchronous intermediate acknowledgment | 105 | Intermediate position for tool:<br>The tools are moved from the source position specified in the job, or from the last acknowledged intermediate position, to the specified target position. |
| Communicating an asynchronous transfer | 201 | Communicate tool movement:<br>The tool is moved from the source position to the specified target position. When moving from a location in a real magazine to an intermediate buffer location, the source location is reserved for the tool.<br>Also observe MD20310, bit 24 (Page 257). |
| | 204 | Communicate magazine position:<br>The magazine location is in the change/load/unload point of the specified target location. |

## Overview of evaluated table parameters

| Acknowledgment status | | 1 | 3 | 7 | 99 | 105 | 201 | 204 |
|---|---|---|---|---|---|---|---|---|
| New tool | Transfer step | x | - | - | - | x | x | x |
| | • from magazine | - | - | - | - | - | xx | xx |
| | • from location | - | - | - | - | - | xx | xx |
| | • to magazine | xx | - | - | - | xx | xx | zz |
| | • to location | xx | - | - | - | xx | xx | zz |
| Old tool | Transfer step | x | - | - | - | x | x | - |
| | • from magazine | - | - | - | - | - | xx | - |
| | • from location | - | - | - | - | - | xx | - |
| | • to magazine | xx | - | - | - | xx | xx | - |
| | • to location | xx | - | - | - | xx | xx | - |

Legend:

- Data item not relevant

x : Number (1…n) of the transfer step from the transfer-step table

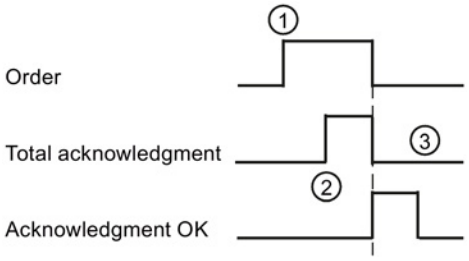
xx: Magazine number, location number of the tool

zz: Magazine number, location number of the load / unload or changing point

The status overview issues the following information:

- Depending on their meaning, the states 1, 6, 105, 201 and 204 should be combined in the acknowledgment table with the transfer steps to make useful acknowledgment steps.
- If status 1 is coded with both transfer step numbers = 0, this acknowledgment step serves as an end acknowledgment of the current status reached via intermediate acknowledgments.
- If a tool has been moved for a real magazine to a buffer (relocate, MVTOOL), acknowledgment with status 6 or total acknowledgment reserves the source location for this tool ($TC_MPP4 bit 1 and bit 2). The behavior is the same as for the removal of a tool from the magazine during tool change. With status 1, there is no reservation of the source location during relocation or MVTOOL.
- States 3 and 7 need only to be coded once in the acknowledgment step table, as no transfers steps are evaluated.
- Status 99 does not need to be coded, it is specified by the "total acknowledgment" bit.

## Typical sequence of any job with total acknowledgment



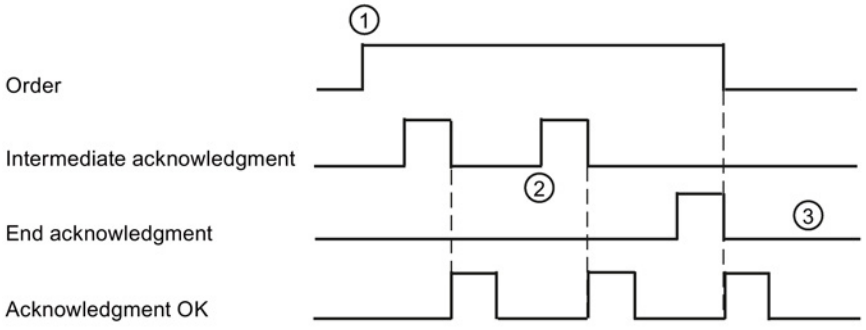

Signal description:

① The PLC user program recognizes from the 0/1 edge of the signal DB43xx.DBX0.0 (job) that a new job has been assigned by the tool management.

② The PLC user program sets the acknowledgment signal in DB42xx.DBX0.0 (total acknowledgment). With activation of the 0/1 edge, the PLC firmware starts transferring the acknowledgment to the tool management.

③ After successful transmission of the acknowledgment to the tool management, the PLC firmware sets a PLC cycle to 1 for the signal "acknowledgment OK" and at the same time the job signal and the acknowledgment bit are reset to 0.

## Typical sequence of any job with total acknowledgment and end acknowledgment




Signal description:

① The PLC user program recognizes from the 0/1 edge of the signal DB43xx.DBX0.0 (job) that a new job has been assigned by the tool management.

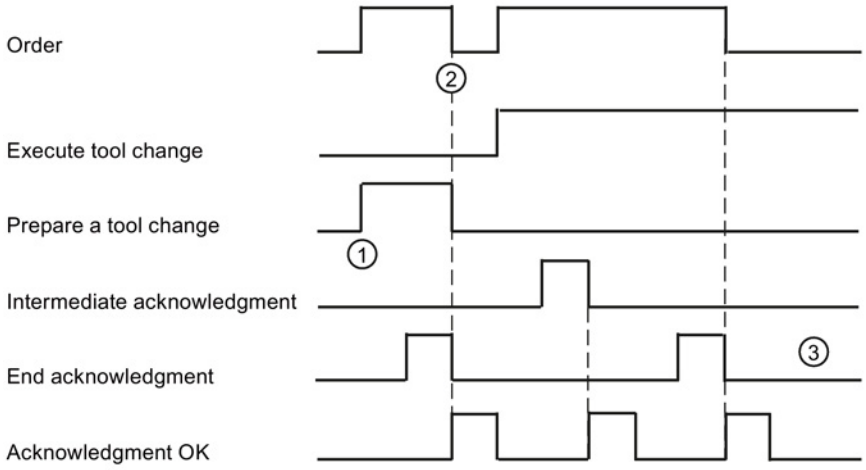
② The PLC user program acknowledges the transfer steps, configured in DB9900, DB9901 and DB9902, with acknowledgment status 105. The tool management updates the tool positions using the transfer steps of the acknowledgments.

③ The execution of the job is acknowledged by the PLC user program with acknowledgment status 1. After successful transmission of the acknowledgment to the tool management, the PLC firmware sets a PLC cycle to 1 for the signal "acknowledgment OK" and at the same time the job signal and the acknowledgment bit are reset to 0.

## Sequence of the tool management with block splitting (machine data setting milling)

```
Txx M6 ;
```

Program L6 is called with M6 (basic setting)




Signal description:

① The PLC user program receives a new job. The job "prepare tool change" and the job "execute tool change" are issued one after the other. Txx and M206 have been programmed in separate NC blocks. Only the job "prepare tool change" is present in the interface in DB43xx. The job "execute tool change" is output only after the end acknowledgment for the job for tool preparation.

② Bit DB43xx.DBX0.0 (job) is reset with acknowledgment of the "prepare tool change" job. If the change command (M206) has already run through the NC main run, the new job is immediately output at the interface.

③ The "execute tool change" job is acknowledged as a normal job. The end acknowledgment OK is returned and the bit for the job is simultaneously reset.

The description of the job ("execute tool change" and "prepare tool change") is not reset. Byte 1 of DB43xx is not overwritten until the next job.

**Note**

**MD20270, MD20310:**

The response of the interface in DB43xx.DBB1 and of NC block processing is affected by the setting in MD20270: $MC_CUTTING_EDGE_DEFAULT and MD20310: $MC_TOOL_MANAGEMENT_MASK bit 5, 6, 7, and 8.

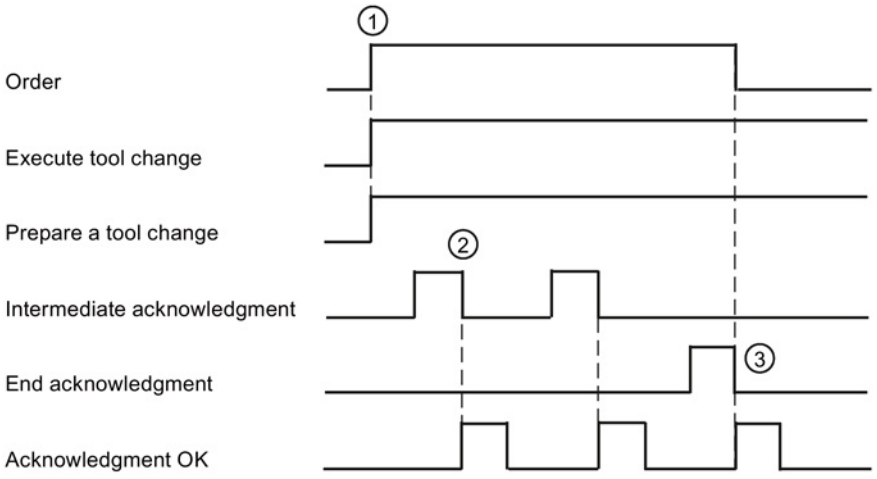
This sequence described here corresponds to the presetting of the machine data.

## Sequence of the tool management without block splitting (machine data setting milling)

```
Txx Myy ;
```

Myy is the setting from MD22560: $MC_TOOL_CHANGE_M_CODE

**This type of programming is not recommended!**




Signal description:

① The PLC user program receives a new job. The job "prepare tool change" and the job "execute tool change" are issued simultaneously. Txx and M206 have been programmed in one NC block.

② Several intermediate steps are acknowledged. The state of the job remains unchanged. The positions of the tools are updated using the transfer steps of the acknowledgments from the tool management.

③ The end acknowledgment OK is returned and the bit for the job is simultaneously reset.

The description of the job ("execute tool change" and "prepare tool change") is not reset. Byte 1 of DB43xx is not overwritten until the next job.

**Note**

**MD20310: $MC_TOOL_MANAGEMENT_MASK**

The response of the interface in DB43xx.DBB1 is affected by the setting in MD20310 bit 10.

This sequence described here corresponds to the presetting of the machine data.

## 9.4.4 Configuring step tables

### Configuring step tables

You can find the transfer-step tables (TM_CTS, DB9900 and TM_VTS, DB9901) and the acknowledgment-step table (TM_ACK, DB9902) in the Programming Tool under "Libraries" → "Special data blocks". The blocks are copied into a project with a double-click.

The structure of the data blocks is permanently fixed.

The blocks are not yet filled with the necessary data. The user must edit them in the Programming Tool, via the menu "View" → "Data block". The constant tables (TM_CTS, DB9900 and TM_ACK, DB9902) are configured by writing the initial data block values in the Programming Tool.

The initial data block values are loaded into the control, along with the PLC user program. Changed initial values are not activated until the PLC is restarted!

### Configuring the transfer steps

Any changes to the tool and magazine positions must be communicated to the tool management by the PLC user program. A table of all mechanical single movements which should be acknowledged / communicated helps with this. For each tool transfer, the table contains the corresponding tool start and target position, or for positioning a magazine location at a transfer point (change, load, unload point), the magazine position and the name of the transfer point.

- The transfer steps 1 … 64 are permanently configured in TM_CTS (DB9900) and can be changed only by reloading.
- The transfer steps 101 … 164 in TM_VTS (DB9901) can be completely or partially overwritten by the PLC user program (e.g. by entering the current magazine location).

### Coding for position from a job

In the constant transfer-step table, the locations of real magazines are not identified by their actual values (e. g. 1/14 for magazine 1 location 14) but with symbolic values (0/1) or (0/2). Otherwise the transfer-step table would be huge for large magazines.

These symbolic values have the following meaning:

| Magazine / Location | Meaning |
|---|---|
| ( 0 / 1 ) | The source position of the only or new tool from the job should be used. |
| ( 0 / 2 ) | The target position of the old tool from the job should be used. |
| ( 0 / 3 ) | The target position of the only tool or new tool from the job should be used. |

This symbolic notation form can only be used for synchronous intermediate and end acknowledgments, since asynchronous messages do not have a job as a reference.

## Example: Constant transfer-step table

| Transfer step | Address DB9900 | Name | Start value | Comment |
|---|---|---|---|---|
| 1 | 0.0 | SrcMag_1 | 0 | Source magazine number of the transfer step |
| | 2.0 | SrcPos_1 | 1 | Source position number of the transfer step |
| | 4.0 | DstMag_1 | 0 | Target magazine number of the transfer step |
| | 6.0 | DstPos_1 | 1 | Target position number of the transfer step |
| 2 | 8.0 | SrcMag_2 | 0 | Source magazine number of the transfer step |
| | 10.0 | SrcPos_2 | 1 | Source position number of the transfer step |
| | 12.0 | DstMag_2 | 9998 | Target magazine number of the transfer step |
| | 14.0 | DstPos_2 | 2 | Target position number of the transfer step |

## Example as a complete step

| Transfer step | of | | to | | Comment |
|---|---|---|---|---|---|
| | Magazine | Location | Magazine | Location | |
| 1 | 0 | 1 | 0 | 1 | Prepare Tool: The magazine is positioned at the changing point of the new tool. |
| 2 | 0 | 1 | 9998 | 2 | Tool change: Tool from magazine to gripper 1 |
| 3 | 9998 | 1 | 9998 | 3 | Tool change: Tool from spindle to gripper 2 |
| 4 | 9998 | 2 | 9998 | 1 | Tool change: Tool from gripper 1 to spindle |
| 5 | 9998 | 3 | 0 | 2 | Tool change: Tool from gripper 2 to magazine |

## Example: Variable transfer-step table

| Transfer step | Address DB9901 | Name | Start value | Comment |
|---|---|---|---|---|
| 101 | 0.0 | SrcMag_101 | 1 | Source magazine number of the transfer step |
| | 2.0 | SrcPos_101 | 0 | Source position number of the transfer step |

| Transfer step | Address DB9901 | Name | Start value | Comment |
|---|---|---|---|---|
| | 4.0 | DstMag_101 | 9998 | Target magazine number of the transfer step |
| | 6.0 | DstPos_101 | 1 | Target position number of the transfer step |
| 102 | 8.0 | SrcMag_102 | 1 | Source magazine number of the transfer step |
| | 10.0 | SrcPos_102 | 0 | Source position number of the transfer step |
| | 12.0 | DstMag_102 | 9998 | Target magazine number of the transfer step |
| | 14.0 | DstPos_102 | 2 | Target position number of the transfer step |

### Example as a complete step

| Transfer step | of | | to | | Comment |
|---|---|---|---|---|---|
| | Magazine | Location | Magazine | Location | |
| 101 | 1 | 0 | 9998 | 1 | "Prepare tool change": The magazine is positioned at the changing point. The source position must be entered by the PLC user program. |
| 102 | 1 | 0 | 9998 | 2 | "Prepare tool change": Tool from magazine to buffer. The source position must be entered by the PLC user program. |

## 9.4.5 Configuring acknowledgment steps

### Configuring the acknowledgment steps

The PLC 31 provides acknowledgment steps to acknowledge tool and magazine movements. These are activated via the corresponding bits on the user interface. The data of these acknowledgment steps (with the exception of special case acknowledgment step 9: total acknowledgment) are stored in acknowledgment step table TM_ACK (DB9902). The transfer steps (number of the transfer step from the transfer step table) for the old and new tool and an acknowledgment status are combined in one acknowledgment status.

It is important to include an acknowledgment step with the status 3 in this table, so that potential errors can be reset. The numbers for the transfer in this acknowledgment step are 0.

### Special meaning of transfer step 0

Only the transfer steps assigned to the acknowledgment step are executed. If only one or no transfer step is assigned, no tool transfer will be carried out for the tool with transfer step = 0. The tool is not available or remains at its original location.

### Example: Acknowledgment step table

| Acknowl-edgment step | Address DB9902 | Name | Start value | Comment |
|---|---|---|---|---|
| 1 | 0.0 | TsNewT_1 | 0 | Transfer step number of the new tool |
| | 1.0 | TsOldT_1 | 0 | Transfer step number of the old tool |
| | 2.0 | State_1 | 3 | Status to NCK |
| 2 | 4.0 | TsNewT_2 | 1 | Transfer step number of the new tool |
| | 5.0 | TsOldT_2 | 0 | Transfer step number of the old tool |
| | 6.0 | State_2 | 1 | Status to NCK |

### Example as a complete step

| Acknowl-edgment step | Transfer step | | Acknowl-edgment status | Comment |
|---|---|---|---|---|
| | New tool | old tool | | |
| 1 | 0 | 0 | 3 | "Cancel order" command. |
| 2 | 2 | 0 | 1 | Prepare Tool: The tool is taken from the magazine (changing point) and placed into gripper 1. |

Acknowledgment is given by setting the corresponding bits in the user interface:

- DB40xx for loading/unloading / relocating or positioning the magazine
- DB42xx for "prepare tool change" and "execute tool change"

After processing, the acknowledgment bit is reset one PLC cycle by the PLC firmware.

In the same data block where the acknowledgments have taken place, for a PLC cycle the feedback message is output in bit 100.0 (acknowledgment OK), or as a static signal in bit 100.1 (acknowledgment error); the error status is output in byte 104 for an acknowledgment error, and in bytes DBB108 to DBB111 the acknowledgment bits last set are output. The user can use these bits to see which acknowledgment step triggered the error. If the PLC user program incorrectly set multiple acknowledgment bits, these are also entered one-to-one in the map. Resetting of the error status occurs when the user acknowledges the error in DB40xx.DBX9.0 or DB42xx.DBX9.0.

## 9.4.6 Adjust the PLC user program

### Adjust the PLC user program

The control of mechanical processes, monitoring and prevention of potential collisions and the acknowledgment of tool position changes are the task of the PLC user program.

### See also

In the PLC project on the Toolbox CD you can find examples of transfer steps and their acknowledgments, for a turning machine with a circular magazine and a milling machine with a chain magazine and dual gripper:

- Application example for milling machine (Page 319)
- Application example for turning machine (Page 306)

These blocks are function examples for the acknowledgment of different tool management jobs.

### Acknowledging jobs

Many tool movements or tool management jobs can be directly acknowledged using total acknowledgment at bit 0.0 of the interface, without previous intermediate acknowledgment.

For example:

- Turns of a circular magazine
- Loading/unloading (only for systems without additional buffers such as handling systems, loaders, etc.)
- Changing manual tools
- Positioning a magazine

### Rules for acknowledgment

If intermediate steps are useful, several rules must be observed during acknowledgment:

The PLC user program must ensure that all acknowledgments are transferred correctly to the tool management.

- Only one acknowledgment signal at a time may be sent to the tool management.
- Synchronous acknowledgments are permissible only for the pending job
- Only valid transfer-step numbers may be used (1 - 64, 100 - 164). For asynchronous messages, at least one transfer step must be input for status 201, for status 204 a transfer step for the new tool must be input.
- Coded positions in the transfer steps may only be used for synchronous acknowledgments and only with the values 0/1, 0/2 or 0/3.
- No illegal acknowledgment states may be used.

- Magazine positioning with a job may only receive a synchronous acknowledgment (end acknowledgment). Intermediate positions must be reported to the tool management with asynchronous messages.
- Acknowledgment signals must be reset by the PLC basic program. After an acknowledgment bit is set, the user interface may not be changed until the feedback message in DB41xx/DB43xx DBB100!
- Asynchronous messages with two transfer steps must be acknowledged on the tool change interface (DB42xx).

## 9.4.7 Information on magazine location

### Overview

It is possible to read up to eight NC variables in one job via the "NC services" interface of DB1200.

### Variable $TC_MPP2 with index 7: Location type

Parameter assignment:

| Read variable from the NC | Address | Signal | Valid values |
|---|---|---|---|
| Job | DB1200.DBX0.0 | Start | 0/1 |
| | DB1200.DBX0.1 | Write variable | 0 |
| | DB1200.DBB1 | Number of variables | 1 … 8 |
| Parameter | DB120x.DBW1000 | Variables index | 7 |
| | DB120x.DBW1002 | Location number | 1 … 31999 |
| | DB120x.DBW1004 | Magazine number | 1 … 9999 |
| Result | DB1200.DBX2000.0 | Job completed | 0/1 |
| | DB1200.DBX2000.1 | Error in job | 0/1 |
| | DB120x.DBX3000.0 | Variable is valid | 0/1 |
| | DB120x.DBB3001 | Access result | 0/3/5/10 |
| | DB120x.DBW3004 | Data from NC variable | n |

Legend:

n > 0: Location type for virtual location

n = 0: "match all" (buffer)

n = 9999: undefined (no virtual location)

## Variable $TC_MPP4 with index 8: Location state

Parameter assignment:

| Read variable from the NC | Address | Signal | Valid values |
|---|---|---|---|
| Job | DB1200.DBX0.0 | Start | 0/1 |
| | DB1200.DBX0.1 | Write variable | 0 |
| | DB1200.DBB1 | Number of variables | 1 … 8 |
| Parameter | DB120x.DBW1000 | Variables index | 8 |
| | DB120x.DBW1002 | Location number | 1 … 31999 |
| | DB120x.DBW1004 | Magazine number | 1 … 9999 |
| Result | DB1200.DBX2000.0 | Job completed | 0/1 |
| | DB1200.DBX2000.1 | Error in job | 0/1 |
| | DB120x.DBX3000.0 | Variable is valid | 0/1 |
| | DB120x.DBB3001 | Access result | 0/3/5/10 |
| | DB120x.DBW3004 | Data from NC variable | n |

Legend:

n = 1 blocked

n = 2 free (<> occupied)

n = 4 reserved for tool in buffer

n = 8 reserved for tool to be loaded

n = 16 occupied in left half location

n = 32 occupied in right half location

n = 64 occupied in top half location

n = 128 occupied in bottom half location

### Variable $TC_MPP6 with index 9: T number of the tool at this location

Parameter assignment:

| Read variable from the NC | Address | Signal | Valid values |
|---|---|---|---|
| Job | DB1200.DBX0.0 | Start | 0/1 |
| | DB1200.DBX0.1 | Write variable | 0 |
| | DB1200.DBB1 | Number of variables | 1 … 8 |
| Parameter | DB120x.DBW1000 | Variables index | 9 |
| | DB120x.DBW1002 | Location number | 1 … 31999 |
| | DB120x.DBW1004 | Magazine number | 1 … 9999 |
| Result | DB1200.DBX2000.0 | Job completed | 0/1 |
| | DB1200.DBX2000.1 | Error in job | 0/1 |
| | DB120x.DBX3000.0 | Variable is valid | 0/1 |
| | DB120x.DBB3001 | Access result | 0/3/5/10 |
| | DB120x.DBW3004 | Data from NC variable | n |

n = T number of the tool on the parameterized location

### Error (for all variables)

In case of error, DB120x.DBX3000.0 = 0 and an entry is made in the access result:

| Values in DB120x.DBB3001 | |
|---|---|
| 0 | No error |
| 3 | Illegal access to object |
| 5 | Invalid address |
| 10 | Object does not exist |

## 9.4.8 PI service: TMMVTL

### Function

With program instance service TMMVTL, it is possible to initiate a job to relocate a tool from the PLC. After an error-free "PI Start," the tool management performs an empty location search in the target magazine for the tool on the defined source location. Subsequently the PLC receives a job for relocating the tool (user interface DB41xx.DBB0).

## Parameter assignment

Start PI service in the NC area

| PI service | Address | Signal | Valid values |
|---|---|---|---|
| Job | DB1200.DBX4000.0 | Start | 0/1 |
| | DB1200.DBB4001 | PI index | 5 |
| Parameter | DB1200.DBW4004 | Tool number<br>(internal T number) | 1 ... 31999 |
| | DB1200.DBW4006 | Source location number | 1 ... 31999 |
| | DB1200.DBW4008 | Source magazine number | 1 ... 31999 |
| | DB1200.DBW4010 | Target location number | -1 |
| | DB1200.DBW4012 | Target magazine number | 1 ... 32000 |
| Result | DB1200.DBX5000.0 | Job completed | 0/1 |
| | DB1200.DBX5000.1 | Error in job | 0/1 |

The tool can be specified either via a T number or by means of the location and magazine numbers. The value -1 is entered at unused specification points.

With the parameter, 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.

For a target magazine number = -1, a search is made in a buffer for the tool corresponding to the assignment obtained from $TC_MDP2.

**Examples:**

- When using buffers to return the tool (for example tool boy 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 searches for a suitable location.
- A tool is to be moved from a background magazine to the front magazine.

# 9.5 Example: Loading/unloading

## Programming

For loading, tools are placed directly in the magazine or the spindle; for unloading, they are removed directly from the magazine. Normally, a single acknowledgment from the operator or the PLC user program (tool holder is closed again) is sufficient as a message that the process is complete. There is no need to configure a transfer step. The total acknowledgment can be set in DB40xx.DBX0.0.

Acknowledgment to tool management:

| Acknowledgment step | Acknowledgment bit | Transfer step new tool | Transfer step old tool | Status |
|---|---|---|---|---|
| xxx | DB4000.DBX0.0 | -- | -- | (99) |

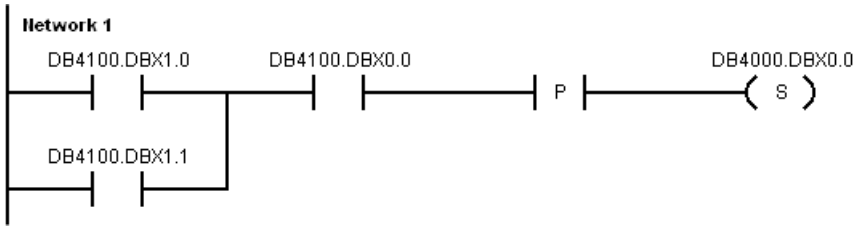



Figure 9-2 Programming in PLC user program

Loading procedures using handling systems or transporting the tool from the spindle into the magazine can be performed using further asynchronous messages.

**Note**

**Manual tools**

Please observe the special issues when using manual tools (Page 255).

A tool that was loaded into the spindle or a magazine, is not a manual tool. As a consequence, it is placed in the magazine using T0/M06.

Additional loading processes are described in the following sections.

### Loading via the spindle with preselection of a magazine location

You can create a new tool directly on the chosen free magazine location or, using the "load" dialog, move a tool from the tool list, which is not located in the magazine, to a magazine location.

1. A job from the tool management is always output at the interface for the loading point. You must acknowledge this job.
2. Then move the tool, which is not yet located in the magazine, into the spindle using Txx M6 or via an asynchronous transfer.
3. Then manually place the tool into the spindle and deposit it in the magazine with T0 M6.

The tools can be oversized or fixed-location coded.

### Loading via the spindle with preselection of a magazine location

You can create a new tool directly on the spindle or, using the "load" dialog, move a tool from the tool list, which is not located in the magazine, to the spindle.

1. A job from the tool management is always output at the interface for the loading point. You must acknowledge this job.
2. Then manually place the tool into the spindle and deposit it in the magazine with T0 M6. A free location is selected by the tool management in which the tool can be deposited.

The tools can be oversized or fixed-location coded.

### Loading directly into the magazine

Position the relevant magazine location next to the loading position. You can create a new tool directly on the chosen free magazine location or, using the "load" dialog, move a tool from the tool list, which is not located in the magazine, to the selected magazine location.

1. A job from the tool management is always output at the interface for the loading point. You must acknowledge this job.
2. Now place the tool in the magazine.

This sequence is not subject to any restrictions or constraints.

# 9.6 Example: Change manual tools

## Programming

In MD22562: $MC_TOOL_CHANGE_ERROR_MODE Bit 1=1, additional tools without magazine allocation have been selected by the NC part program. The selected tool must be inserted in the machine manually and removed again manually after machining ("manual tools").

The operator must ensure that the data block for the tool on the spindle is in the NCK, or that he/she puts the appropriate tool onto the spindle for the data block stored in the NCK.

**Note**

The responsibility is on the user to comply with the safety regulations via the PLC user program.

The PLC user program is informed with DB43xx.DBX1.5 and DBX1.6 whether a manual tool is involved in a tool change job. With alarm 17212: "Channel %1, Manual tool %2, Duplo No. %3, Load to tool holder %4" or Alarm 17214: "Remove manual tool from spindle/tool holder", the operator is requested to execute the tool change.

The alarms are reset after a tool change by the acknowledgment from the PLC.

## Starting position 1

**The manual tool in the spindle should be exchanged for another manual tool.**

Job from tool management to the PLC user program (tool change):

DB4300.DBX0.0, DBX1.2, DBX1.5 and DBX1.6 ("Prepare tool change")

| | | |
|---|---|---|
| DB43xx.DBW6 | Source magazine number | 9999 |
| DB43xx.DBW8 | Source location number | 1 |
| DB43xx.DBW10 | Target magazine number | 9999 |
| DB43xx.DBW12 | Target location number | 1 |

Acknowledgment "Prepare tool change":

| Acknowledg-ment step | Acknowledgment bit | Transfer step for the new tool | Transfer step for the old tool | Status |
|---|---|---|---|---|
| xxx | DB4200.DBXx.x | 0 | 0 | 1 |

Displayed on the interface: DB4300.DBX0.0 /1.1, DBX1.5 and DBX1.6 ("Execute")

The job remains otherwise unchanged, the tools are still at the starting positions. An intermediate step can be inserted for acknowledging the removal of the old tool:

## Synchronous acknowledgment

The tool is no longer in the spindle.

| Transfer step | From | | To | | Comment |
|---|---|---|---|---|---|
| | Magazine | Location | Magazine | Location | |
| 6<br>DB9900.DBW40 | 9998 | 1 | 9999 | 1 | Tool removed from the spindle |

| Acknowledgment step | Acknowledgment bit | Transfer step for the new tool | Transfer step for the old tool | Status |
|---|---|---|---|---|
| xxx | DB4200.DBXx.x | 0 | 6 | 105 |

After intermediate acknowledgment of the empty spindle and insertion of the new tool in the spindle, the tool change is terminated with a total acknowledgment:

| Acknowledgment step | Acknowledgment bit | Transfer step for the new tool | Transfer step for the old tool | Status |
|---|---|---|---|---|
| | DB4200.DBX0.0 | | | (99) |

## Starting position 2

**A manual tool is in the spindle and is to be replaced by a tool from the magazine.**

Job from tool management to the PLC user program (tool change):

DB4300.DBX0.0, DBX1.2 and DBX1.6 ("Prepare tool change")

| | | |
|---|---|---|
| DB43xx.DBW6 | Source magazine number | 1 |
| DB43xx.DBW8 | Source location number | 6 |
| DB43xx.DBW10 | Target magazine number | 9999 |
| DB43xx.DBW12 | Target location number | 1 |

Acknowledgment "Prepare tool change":

| Acknowledgment step | Acknowledgment bit | Transfer step for the new tool | Transfer step for the old tool | Status |
|---|---|---|---|---|
| xxx | DB4200.DBXx.x | 0 | 0 | 1 |

Displayed on the interface: DB4300.DBX0.0 /1.1 and 1.6 ("Execute tool change")

The job remains otherwise unchanged, the tools are still at the starting positions.

## Synchronous acknowledgment

The old tool is no longer in the spindle.

| Transfer step | From | | To | | Comment |
|---|---|---|---|---|---|
| | Magazine | Location | Magazine | Location | |
| 6<br>DB9900.DBW40 | 9998 | 1 | 9999 | 1 | Asynchronous message, unload the tool from the spindle |

| Acknowledgment step | Acknowledgment bit | Transfer step for the new tool | Transfer step for the old tool | Status |
|---|---|---|---|---|
| xxx | DB4200.DBXx.x | 0 | 6 | 105 |

The spindle is now empty, the old tool is outside the magazine.

## Next step

Synchronous acknowledgment, new tool to gripper 1

| Transfer step | From | | To | | Comment |
|---|---|---|---|---|---|
| | Magazine | Location | Magazine | Location | |
| 3<br>DB9900.DBW16 | 0 | 1 | 9998 | 2 | New tool to gripper 1 |

| Acknowledgment step | Acknowledgment bit | Transfer step for the new tool | Transfer step for the old tool | Status |
|---|---|---|---|---|
| xxx | DB4200.DBXx.x | 3 | 0 | 105 |

The job is unchanged.

## Next step

Synchronous acknowledgment of new tool from gripper 1 to spindle

| Transfer step | From | | To | | Comment |
|---|---|---|---|---|---|
| | Magazine | Location | Magazine | Location | |
| 4<br>DB9900.DBW24 | 9998 | 2 | 9998 | 1 | New tool from gripper 1 to spindle |

| Acknowledgment step | Acknowledgment bit | Transfer step for the new tool | Transfer step for the old tool | Status |
|---|---|---|---|---|
| xxx | DB4200.DBXx.x | 4 | 0 | 105 |

This completes the tool movement.

## End acknowledgment

| Acknowledgment step | Acknowledgment bit | Transfer step for the new tool | Transfer step for the old tool | Status |
|---|---|---|---|---|
| xxx | DB4200.DBXx.x | 0 | 0 | 1 |

The step New tool from gripper 1 to spindle can be omitted and replaced by a total acknowledgment. This also informs the tool management that all tools are in their target positions.

| Acknowledgment step | Acknowledgment bit | Transfer step for the new tool | Transfer step for the old tool | Status |
|---|---|---|---|---|
| | DB4200.DBX0.0 | | | (99) |

# 9.7 Application example for turning machine

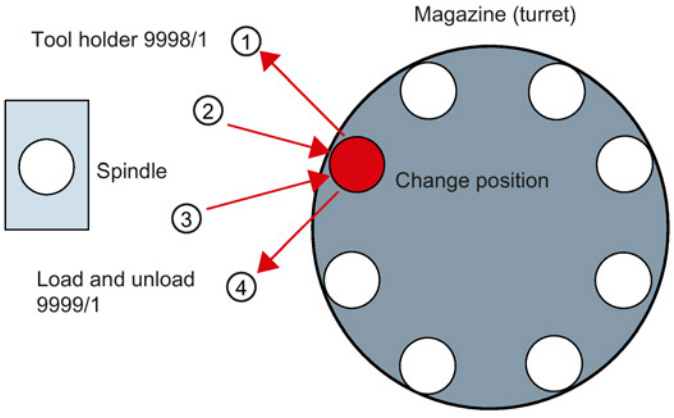
## 9.7.1 Example: Turning machine with revolver magazine (MAGKONF_MPF)

### Example file

You can find the program for configuring the magazine in the Toolbox.

The program can be read into the control and should be adjusted for the specific machine concerned.

### Configuration




1 ... 4 Transfer steps

Figure 9-3 Turning machine with revolver magazine

### Description of the program

To start with, all the old magazine definitions and tools are deleted. As the program sequence continues, all magazines and buffers are newly created and assigned by writing the magazine parameters.

Search strategies for tools and magazine locations can be defined in N70.

For revolver magazines it is advisable to define all locations as fixed-location coded. For magazine type 3 N320 $TC_MAP3[NUM_MAG] = 81 (Bit6 = 1) is set.

The chain magazine locations are set up from N430 to N500. Magazine location type = 0 means that the magazine location can be loaded with tools of different location types.

The buffers are set up from N520 onwards.

From N920, the buffers are assigned to the spindle / tool holder and the magazine.

After the magazine configuration program has finished running, restart the NC using (NCK reset).

## See also

You will find a precise description of the parameters used in the Tool Management Function Manual of SINUMERIK 840D sl.

## Example MAG_CONF_MPF

1. Plant configuration:
   - 1 revolver magazine with 8 locations (can be set up in N40)
   - 1 loading point
   - 3 buffer locations (can be set up in N50, assignments from N540)
2. Part program:

```
;MAG_CONF_MPF
N10 def int NUM_MAG,MAG_TYPE, LOCATIONS,
PLACE, NUM_BUFFER, NUM_LOAD, PLACE_SEARCH
;
N20 NUM_MAG = 1                                ;Number of the magazine
N30 MAG_TYPE = 3                               ;Magazine type (1: chain,
                                               3: revolver, 5: box magazine)
N40 LOCATIONS = 8                              ;Number of magazine locations
N50 NUM_BUFFER = 1                             ;Number of buffers (spindle, grip-
                                               per)
N60 NUM_LOAD = 1                               ; Number of loading points
N70 PLACE_SEARCH = 257                         ; Type of search strategy
;= 257 Bit13=0 no exchange of the old tool on the location of the new tool
;Setting for pickup magazine
;=12289 Bit13=1 exchange old tool on the location of the new tool
;Setting for chain magazine
N80;
N90;
;Check parameters
N100 STOPRE
N110 if ((NUM_MAG==0)or(LOCATIONS==0))
N120 Err1:STOPRE
N130 MSG("Wrong Parameter --> Cancel")
N140 G04 F4
N150 STOPRE
N160 M0
N170 GOTOB Err1
N180 endif
N190; Magazine configuration
N200;
N210;
```

```
N220; Delete old data when magazine 1 is set up
N230 if NUM_MAG ==1
N240 $TC_MAP1[0]=0                              ; Delete magazine
N250 $TC_DP1[0,0]=0                             ; Delete Tools
N260 STOPRE
N270 endif
; Configuration
;
N280 $TC_MAMP2= PLACE_SEARCH                    ; Type of search strategy
;
; Magazine
; Set up magazine
N290 $TC_MAP1[NUM_MAG]= MAG_TYPE
N300 $TC_MAP2[NUM_MAG]="MAGAZINE"<<NUM_MAG
N310 if MAG_TYPE == 3
N320 $TC_MAP3[NUM_MAG]=81                       ; Magazine status, all locations
                                                fixed-location coded for revolver
                                                magazine
N330 else
N340 $TC_MAP3[NUM_MAG]=17                       ; Magazine status
N350 endif
N360 $TC_MAP4[NUM_MAG]=-1
N370 $TC_MAP5[NUM_MAG]=-1
N380 $TC_MAP6[NUM_MAG]=1                        ; Number of lines in magazine
N390 $TC_MAP8[NUM_MAG]=0
N400 $TC_MAP9[NUM_MAG]=0
N410 $TC_MAP7[NUM_MAG]=LOCATIONS                ; Number of magazine locations
N420 $TC_MAP10[NUM_MAG]=PLACE_SEARCH
;
; Magazine locations
N430 for PLACE=1 to LOCATIONS
N440 STOPRE
N450 $TC_MPP1[NUM_MAG,PLACE]=1                  ; Location type
N460 $TC_MPP2[NUM_MAG,PLACE]=0                  ; Location type, 0 compatible with
                                                every tool location type
N470 $TC_MPP3[NUM_MAG,PLACE]=1                  ; Consider adjacent location On (Off
                                                would be 0)
N480 $TC_MPP4[NUM_MAG,PLACE]=2                  ; Location status
N490 $TC_MPP5[NUM_MAG,PLACE]=PLACE              ; Location type index
N500 endfor
N510 STOPRE
;
N520; Definition of buffer magazine (always number 9998)
;
```

```
N530 $TC_MAP1[9998]=7                          ; Magazine type 7: Buffer
N540 $TC_MAP2[9998]="BUFFER"<<NUM_MAG
N550 $TC_MAP3[9998]=17                         ; Magazine status
N560 $TC_MAP6[9998]=1                          ; Number of lines
N570 $TC_MAP7[9998]=NUM_BUFFER                 ; Number of locations
;
; Locations in the buffer
;Spindle
N580 $TC_MPP1[9998,1]=2                        ; Location type (here spindle)
N590 $TC_MPP2[9998,1]=0                        ; Location type (here always 0)
N600 $TC_MPP3[9998,1]=0                        ; Consider adjacent location Off
N610 $TC_MPP4[9998,1]=2                        ; Location status
N620 $TC_MPP5[9998,1]=1                        ; Location type index
;
N630; Gripper
N640 FOR PLACE=2 to NUM_BUFFER
N650 STOPRE
N660 $TC_MPP1[9998,PLACE]=3                    ;(here gripper)
N670 $TC_MPP2[9998,PLACE]=0                    ;(here always 0)
N680 $TC_MPP3[9998,PLACE]=0                    ; Consider adjacent location Off
N690 $TC_MPP4[9998,PLACE]=2                    ; Location status
N700 $TC_MPP5[9998,PLACE]=PLACE                ; Location type index
N710 endfor
N720 STOPRE
;
;
N730; Definition of loading magazine (always number 9999)
;
N740 $TC_MAP1[9999]=9                          ; Magazine type 9: Loading magazine
N750 $TC_MAP2[9999]="LOADING
MAGAZINE"<<NUM_MAG
N760 $TC_MAP3[9999]=17                         ; Magazine status
N770 $TC_MAP4[9999]=-1
N780 $TC_MAP5[9999]=-1
N790 $TC_MAP6[9999]=1                          ; Number of lines
N800 $TC_MAP7[9999]=NUM_LOAD                   ; Number of locations
N810 STOPRE;
;
N820; Loading magazine locations
;
N830 for PLACE=1 to NUM_LOAD
N840 STOPRE
N850 $TC_MPP1[9999,PLACE]=7                    ; Location type Loading point
N860 $TC_MPP2[9999,PLACE]=0                    ; Location type (here always 0)
```

```
N870 $TC_MPP3[9999,PLACE]=0                    ; Consider adjacent location Off
N880 $TC_MPP4[9999,PLACE]=2                    ; Location status
N890 $TC_MPP5[9999,PLACE]=PLACE                ; Location type index
N900 endfor
N910 STOPRE
;
;
N920; Offsets (clearances) ; Clearances to magazine
;
; Buffer
N930 for PLACE=1 to NUM_BUFFER
N940 $TC_MDP2[1,PLACE]=0
N950 endfor
N960 STOPRE
;
;Loading points
N970 for PLACE=1 to NUM_LOAD
N980 stopre
N990 $TC_MDP1[1,PLACE]=0
N1000 endfor

N1010 M30                                      ; End
```

### Display in operating software

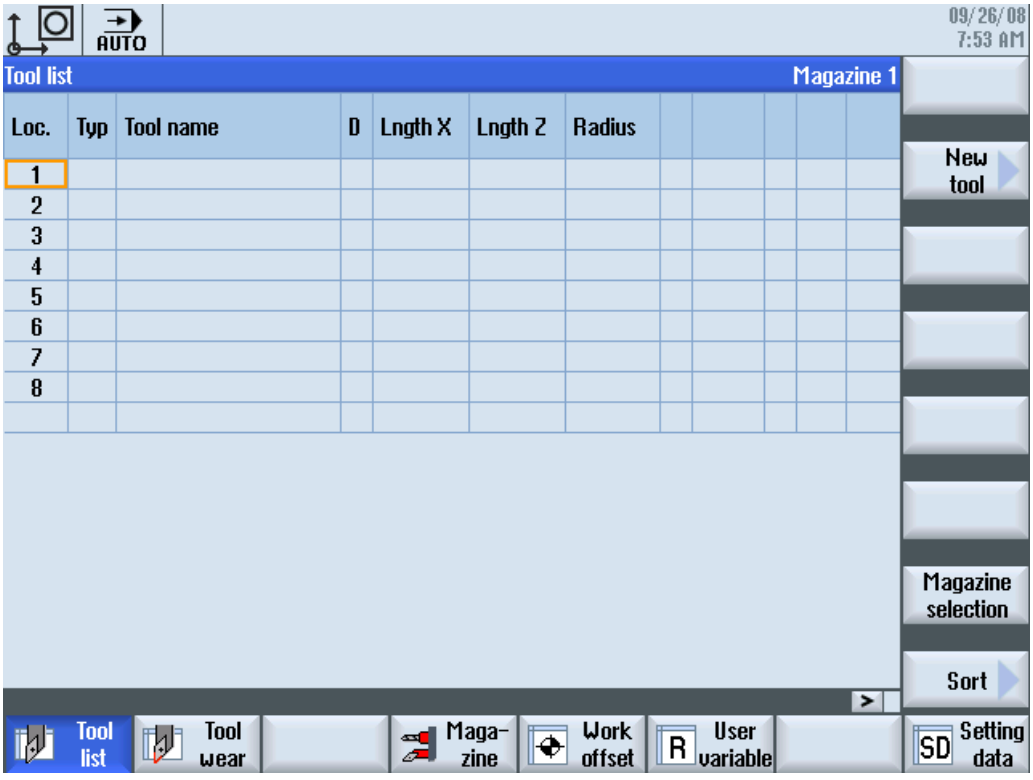



Figure 9-4 Turning machine tool list

## 9.7.2 Example: Acknowledgment steps (turning machine)

### Acknowledgment steps

Generally, the mechanical sequences on a turning machine are simpler than those for a milling machine. In the configuration described in the previous chapter without additional buffer, tool changes can be acknowledged after the mechanical movements without transfer steps. The PLC user program must detect incoming jobs in the job interfaces and execute the mechanical movements.

Total acknowledgment to tool management:

| Acknowledgment step | Acknowledgment bit | Transfer step new tool | Transfer step old tool | Status |
|---|---|---|---|---|
| -- | DB4000.DBX0.0 | -- | -- | (99) |
| -- | DB4200.DBX0.0 | -- | -- | (99) |

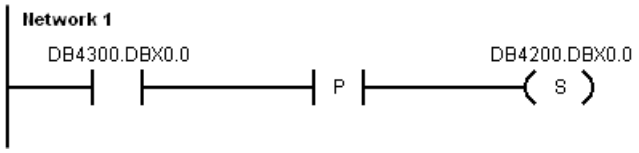



Figure 9-5 Programming in PLC user program

An asynchronous message can be used for magazine movements without a job from the tool management.

Acknowledgment to tool management:

| Acknowledgment step | Acknowledgment bit | Transfer step new tool | Transfer step old tool | Status |
|---|---|---|---|---|
| xxx | DB4000.DBXx.x | 101 | 0 | 204 |

| Transfer step | Source | | Objective | | Comment |
|---|---|---|---|---|---|
| | Maga-zine | Location | Maga-zine | Location | |
| 101<br>DB9901.DBW0 | 1 | n | 9998 | 1 | The variable location in Magazine 1 is positioned at the changing point to the spindle. |

n is the actual location number (n ≠ 0) to be entered by the PLC user program in the variable transfer table.

## 9.7.3 Example: Tool change cycle for turning machine

### Transfer variables of the T replacement cycle:

The "Replacement tools for tool management" option is not active.

Prerequisite is that the tool management has been informed of each magazine movement.

With MD10717: $MN_T_NO_FCT_CYCLE_NAME (name of cycle to be executed instead of the T function, e.g. "TCHANGE"), you can set that a cycle is run through when the T command is called. The cycle is run through at every T call, irrespective of whether a new tool or the already active tool is called. The position of the turret can be set on the programmed tool in this cycle (POSM). This is necessary when the turret has been positioned manually after a tool selection and the tool has not been deselected. In this case, the NC does not issue a new job for tool change at the interface.

## Programming

In the programming example, a job for positioning the magazine with POSM is output at the interface. During the magazine positioning, the block processing of the NC must be controlled by the PLC user program. In most cases, it is useful to set the load or feed disable during this time. As soon as the position setpoint specified in the job is reached (asynchronous message of the magazine position), the job is terminated with total acknowledgment.

```
PROC L6 SAVE SBLOF DISPLOF
  IF $C_T_PROG==1                    ; T is numeric
    IF $C_T==0                       ; T=0
      T=0
    ENDIF
    IF $C_T>0
      IF $C_T<=$TC_MAP7[1]           ; Does the magazine location exist?
      POSM($C_T)                     ; Position magazine
      ENDIF
      T=$C_T                         ; T programming of location number
    ENDIF
  ENDIF

  IF $C_TS_PROG==1                   ; T is an identifier
    _TNO_NEW=GETT($C_TS,1)           ; Querying of T number
    IF _TNO_NEW>0                    ; Does the T number exist?
      _TL_NEW=$A_MYMLN[_TNO_NEW]     ; Querying of location number
    ENDIF
    IF _TL_NEW>0                     ; Is the tool in the magazine?
      POSM(_TL_NEW)                  ; Position magazine
    ENDIF
    T=$C_TS                          ; T programming without address extension
  ENDIF
M17
```

## Tool call for tool measuring

For special applications, such as measuring in JOG, it is necessary to load a specific tool (e.g. miller_15, duplo number 2) - independent of its status - onto the spindle or the tool holder; this can be the case, for example, if a tool was being used, but was locked by the tool monitoring function. Now, before being released again, it should be measured. The NC command TCA is available to call a tool independent of the status and by selecting duplo number and tool holder number; this is used when measuring tools in JOG. The NC command TCA ("Tool name", duplo no., tool holder no.) behaves like the T command with respect to alarm and command output to the PLC.

## Supplementary conditions

The following supplementary conditions must be observed for an active T function replacement (default setting for turning):

- The cycle defined in MD10717 $MN_T_NO_FCT_CYCLE_NAME and executed with the T call, is not started with the TCA command.

  To ensure that a tool change cycle can also be started when TCA is programmed, the language command TCA must be redefined in _TCA (default setting NC).

- TCA can be substituted with a replacement cycle. The name of the replacement cycle is entered in MD15710 $MN_TCA_CYCLE_NAME. This replacement cycle is called from the cycle of the redefined TCA function (TCA.SPF from the toolbox).
- The T and TCA replacement can be implemented in a common replacement cycle.

There is a sample cycle in the toolbox for the replacement of both tool calls in a common replacement cycle.

**Note**

The offset selection, in accordance with MD20270 $MC_CUTTING_EDGE_DEFAULT, acts in the same way as for the T command. TCA and D must not be programmed in the same block.

The following example describes the sequence:

```
PROC TCA(STRING[64] _TOOL_NAME,INT _DUPLO,INT _TH_NO)
                                          ;VERSION: 01.00.07. Oct 22, 2009
                                          ;CHANGE: 01.00.07. Oct 22, 2009
TCA(_TOOL_NAME,_DUPLO,_TH_NO)             ;Tool call to NC
; here, insert the machine functions for the change
M17
```

## References

The function is available irrespective of tool management and is described in full in:

Function Manual, Basic Functions (K1) in Section "Replacement of functions by subprograms"

## 9.7.4 Example: Block search and MCALL

### Requirement

In the tool change subprogram, you must ensure for the block search via modally called cycles (e.g. drilling cycles) that the modal call is retained and ensure that the correct tool is used.

After a block search with calculation, the tool change subprogram (L6) is called in the block search ASUB. The modal call may only be deselected there when L6 has not been called from the ASUB.

### Programming example

```
L6

N100 IF $P_MC == TRUE   ; MCALL is active

N110   IF $AC_ASUP <> 0  ; Call is performed via search ASUB or overstore → do not
select MCALL

N140   ELSE

N150    MCALL

N160 ENDIF

N170 ENDIF
```

The machine manufacturer must also ensure that a real tool change with axis movements and a modal cycle call are permitted. Under certain circumstances, the tool change must be prevented when the programmed tool after a block change is not the same as the tool in the spindle.

### References

SINUMERIK Operate 828D Commissioning Manual, Section "Technological cycles for drilling"

## 9.7.5 Example: Turning machine with counterspindle

### Magazine configuration

In the magazine configuration the magazine is assigned twice as many locations as actually exist, e.g. in the case of revolver with 12 locations, 24 locations are set up. Location 1-12 for the main spindle, location 13-24 for the counterspindle.

The user program positions the magazine in such a way that, for example, the same position is approached for location 1 and location 13. So each real magazine location corresponds to a virtual magazine location for the main spindle and a virtual magazine location for the counterspindle.

## 9.7.6 Example: Test for empty buffer

### Procedure

Read the T number of a tool in gripper 1 and 2:

1. In the PLC user program, enter the parameters in DB1200.
2. In DB1200.DBX0.0, set the start for reading the location states.

Once the order has been successfully executed, the results are entered starting from DB1200.DBB3000:

| Read variable from the NCK | Address | Signal | Values |
|---|---|---|---|
| Parameters | DB1200.DBW1000 | Variables index | 9 |
| | DB1200.DBW1002 | Location number | 2 |
| | DB1200.DBW1004 | Magazine number | 9998 |
| | DB1201.DBW1000 | Variables index | 9 |
| | DB1201.DBW1002 | Location number | 3 |
| | DB1201.DBW1004 | Magazine number | 9998 |
| Order | DB1200.DBX0.1 | Write variable | 0 |
| | DB1200.DBX0.2 | PI service | 0 |
| | DB1200.DBB1 | Number of variables | 2 |
| | DB1200.DBX0.0 | Start | → 1 |
| Result | DB1200.DBB2000.0 | Order completed | 1 |
| | DB1200.DBX2000.1 | Error in order | 0 |
| | DB1200.DBX3000.0 | Variable is valid | 1 |
| | DB1200.DBB3001 | Access result | 0 |
| | DB1200.DBW3004 | Data from NCK variable | n |
| | DB1201.DBX3000.0 | Variable is valid | 1 |
| | DB1201.DBB3001 | Access result | n |
| | DB1201.DBW3004 | Data from NCK variable | 0 |

## 9.7.7 Example: Transporting a tool from a buffer into the magazine

### Procedure

A tool is to be moved from a buffer (for example, of a gripper) into the magazine. The empty location search for the tool from gripper 1 (magazine 9998, location 2) is executed with PI Service TMMVTL and an order to relocate the tool is generated.

In the PLC user program, enter the parameters in DB1200, and in DB1200.DBX4000.0, set the start signal of the PI service.

| Start PI Services in the NCK area | Address | Signal | Values |
|---|---|---|---|
| Parameter | DB1200.DBW4004 | Tool number | 0 |
| | DB1200.DBW4006 | Source location number | 2 |
| | DB1200.DBW4008 | Source magazine number | 9998 |
| | DB1201.DBW4010 | Target location number | -1 |
| | DB1201.DBW4012 | Target magazine number | 1 |
| Job | DB1200.DBB4001 | PI index | 5 |
| | DB1200.DBX4000.0 | Start | → 1 |
| Result | DB1200.DBB5000.0 | Job completed | 1 |
| | DB1200.DBX5000.1 | Error in job | 0 |

DB410x.DBX0.0 delivers an order to relocate the tool from the buffer. The target location in magazine 1 is in DB4100.DBW12. With it, the PLC user program can execute the necessary sequence.

## 9.7.8 Example: Repeat "Prepare tool change" order

### Sequence: Repeat the command

For a milling machine with toolboy and shifter, the asynchronous transfer message and order can cross:

- The toolboy was acknowledged to the tool management as the target position for the old tool with the end acknowledgment.
- The part program is continued with the new tool and after a very short time needs the previous (old) tool once more.
- The tool management generates the next preparation order for the tool change with the toolboy as the source position for what will then be the new tool.
- At the same time the PLC user program transfered the tool from the toolboy into the shifter.
- The PLC user progamm communicates the tool movement from the toolboy to the shifter asynchronously and starts onward transport to the magazine.
- In the next cycle the AWP recognizes the new order to relocate the tool from the toolboy to the spindle.

But the tool is no longer in the toolboy! The PLC user program must detect such a condition (monitoring: does toolboy or shifter contain a tool?). It can now cancel the return of the tool into the magazine. A repeat order (Status 7) can then be requested from the tool management:

- Meanwhile, the tool management has received the message that the tool is in the shifter.
- It generates a new preparation order with the source position shifter for the new tool.

**Note**

The acknowledgment to repeat the order "Prepare tool change" may only be issued **before** the end acknowledgment of the "Prepare tool change" order!

Acknowledgment to tool management:

| Acknowledgment step | Acknowledgment bit | Transfer step new tool | Transfer step old tool | Status |
|---|---|---|---|---|
| xxx | DB4000.DBXx.x | 0 | 0 | 7 |

# 9.8 Application example for milling machine

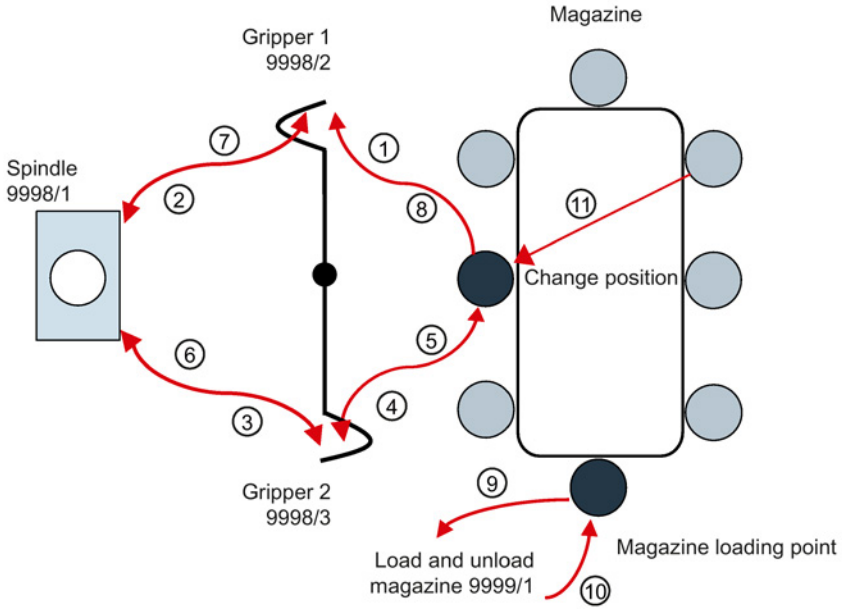
## 9.8.1 Example: Milling machine with chain magazine and dual gripper (MAGKONF_EXAMPLE_MPF)

### Example file

You can find the program for configuring the magazine in the Toolbox.

The program can be read into the control and should be adjusted for the specific machine concerned.

### Configuration




1 ... 11 Transfer steps

Figure 9-6 Milling machine with chain magazine

Description of the buffers and loading points:

| Magazine | Location | Meaning |
|---|---|---|
| 1 | xx | Real magazine (chain, plate, box), position xx |
| 9998 | 1 | Spindle |
| 9998 | 2 | Gripper 1 |
| 9998 | 3 | Gripper 2 |
| 9999 | 1 | Magazine loading point |

## Description of the program

To start with, all the old magazine definitions and tools are deleted. As the program sequence continues, all magazines and buffers are newly created and assigned by writing the magazine parameters.

Search strategies for tools and magazine locations can be selected in N70. Here it is determined whether the tool can be placed directly in the new tool location, when changing out of the spindle. This makes the tool change possible in a single mechanical sequence, so shorter changeover times can be achieved. This sequence is not possible for pickup magazines.

The chain magazine locations are set up from N430 to N500. Magazine location type = 0 means that the magazine location can be loaded with tools of different location types.

The buffers are set up from N520 onwards. If additional buffers are available (toolboy, shifter...), the number should be changed in N50.

Additionally available loading points should be dealt with in the same way in N60.

From N920, the buffers are assigned to the spindle / tool holder and the magazine.

After the magazine configuration program has finished running, restart the NC (NCK reset).

## See also

Further information:

- The configuration of a disk-type magazine with fixed-location coding is performed as for the configuration of a revolver magazine: Example: Turning machine with revolver magazine (MAGKONF_MPF) (Page 306)
- You will find a precise description of the parameters used in the Tool Management Function Manual of SINUMERIK 840D sl.

## Example MAG_CONF_MPF

1. Plant configuration:
   - 1 chain magazine with 8 locations (can be set up in N40)
   - 1 loading point
   - 3 buffer locations (can be set up in N50, assignments from N540)
2. Part program:

```
;MAG_CONF_MPF
N10 def int NUM_MAG,MAG_TYPE, LOCATIONS,
PLACE, NUM_BUFFER, NUM_LOAD, PLACE_SEARCH
;
N20 NUM_MAG = 1                                ;Number of the magazine
N30 MAG_TYPE = 1                               ;Magazine type (1: chain,
                                               3: revolver, 5: box magazine)
N40 LOCATIONS = 8                              ;Number of magazine locations
```

```
N50 NUM_BUFFER = 3                              ;Number of buffers (spindle, gripper)
N60 NUM_LOAD = 1                                ; Number of loading points
N70 PLACE_SEARCH = 12289                        ; Type of search strategy
;= 257 Bit13=0 no exchange of the old tool on the location of the new tool
;Setting for pickup magazine
;=12289 Bit13=1 exchange old tool on the location of the new tool
;Setting for chain magazine
N80;
N90;
;Check parameters
N100 STOPRE
N110 if ((NUM_MAG==0)or(LOCATIONS==0))
N120 Err1:STOPRE
N130 MSG("Wrong Parameter --> Cancel")
N140 G04 F4
N150 STOPRE
N160 M0
N170 GOTOB Err1
N180 endif
N190; Magazine configuration
N200;
N210;
N220; Delete old data when magazine 1 is set up
N230 if NUM_MAG ==1
N240 $TC_MAP1[0]=0                              ; Delete magazine
N250 $TC_DP1[0,0]=0                             ; Delete Tools
N260 STOPRE
N270 endif
; Configuration
;
N280 $TC_MAMP2= PLACE_SEARCH                    ; Type of search strategy
;
; Magazine
; Set up magazine
N290 $TC_MAP1[NUM_MAG]= MAG_TYPE
N300 $TC_MAP2[NUM_MAG]="MAGAZINE"<<NUM_MAG
N310 if MAG_TYPE == 3
N320 $TC_MAP3[NUM_MAG]=81                       ; Magazine status, all locations fixed-location coded for revolver magazine
N330 else
N340 $TC_MAP3[NUM_MAG]=17                       ; Magazine status
N350 endif
```

```
N360 $TC_MAP4[NUM_MAG]=-1
N370 $TC_MAP5[NUM_MAG]=-1
N380 $TC_MAP6[NUM_MAG]=1                        ; Number of lines in magazine
N390 $TC_MAP8[NUM_MAG]=0
N400 $TC_MAP9[NUM_MAG]=0
N410 $TC_MAP7[NUM_MAG]=LOCATIONS                ; Number of magazine locations
N420 $TC_MAP10[NUM_MAG]=PLACE_SEARCH
;
; Magazine locations
N430 for PLACE=1 to LOCATIONS
N440 STOPRE
N450 $TC_MPP1[NUM_MAG,PLACE]=1                  ; Location type
N460 $TC_MPP2[NUM_MAG,PLACE]=0                  ; Location type, 0 compatible with
                                                every tool location type
N470 $TC_MPP3[NUM_MAG,PLACE]=1                  ; Consider adjacent location On (Off
                                                would be 0)
N480 $TC_MPP4[NUM_MAG,PLACE]=2                  ; Location status
N490 $TC_MPP5[NUM_MAG,PLACE]=PLACE              ; Location type index
N500 endfor
N510 STOPRE
;
N520; Definition of buffer magazine (always number 9998)
;
N530 $TC_MAP1[9998]=7                           ; Magazine type 7: Buffer
N540 $TC_MAP2[9998]="BUFFER"<<NUM_MAG
N550 $TC_MAP3[9998]=17                          ; Magazine status
N560 $TC_MAP6[9998]=1                           ; Number of lines
N570 $TC_MAP7[9998]=NUM_BUFFER                  ; Number of locations
;
; Locations in the buffer
;Spindle
N580 $TC_MPP1[9998,1]=2                         ; Location type (here spindle)
N590 $TC_MPP2[9998,1]=0                         ; Location type (here always 0)
N600 $TC_MPP3[9998,1]=0                         ; Consider adjacent location Off
N610 $TC_MPP4[9998,1]=2                         ; Location status
N620 $TC_MPP5[9998,1]=1                         ; Location type index
;
N630; Gripper
N640 FOR PLACE=2 to NUM_BUFFER
N650 STOPRE
N660 $TC_MPP1[9998,PLACE]=3                     ;(here gripper)
N670 $TC_MPP2[9998,PLACE]=0                     ;(here always 0)
N680 $TC_MPP3[9998,PLACE]=0                     ; Consider adjacent location Off
N690 $TC_MPP4[9998,PLACE]=2                     ; Location status
```

```
N700 $TC_MPP5[9998,PLACE]=PLACE                ; Location type index
N710 endfor
N720 STOPRE
;
;
N730; Definition of loading magazine (always number 9999)
;
N740 $TC_MAP1[9999]=9                          ; Magazine type 9: Loading magazine
N750 $TC_MAP2[9999]="LOADING
MAGAZINE"<<NUM_MAG
N760 $TC_MAP3[9999]=17                         ; Magazine status
N770 $TC_MAP4[9999]=-1
N780 $TC_MAP5[9999]=-1
N790 $TC_MAP6[9999]=1                          ; Number of lines
N800 $TC_MAP7[9999]=NUM_LOAD                   ; Number of locations
N810 STOPRE;
;
N820; Loading magazine locations
;
N830 for PLACE=1 to NUM_LOAD
N840 STOPRE
N850 $TC_MPP1[9999,PLACE]=7                    ; Location type Loading point
N860 $TC_MPP2[9999,PLACE]=0                    ; Location type (here always 0)
N870 $TC_MPP3[9999,PLACE]=0                    ; Consider adjacent location Off
N880 $TC_MPP4[9999,PLACE]=2                    ; Location status
N890 $TC_MPP5[9999,PLACE]=PLACE                ; Location type index
N900 endfor
N910 STOPRE
;
;
N920; Offsets (clearances) ; Clearances to magazine
;
; Buffer
N930 for PLACE=1 to NUM_BUFFER
N940 $TC_MDP2[1,PLACE]=0
N950 endfor
N960 STOPRE
;
;Loading points
N970 for PLACE=1 to NUM_LOAD
N980 stopre
N990 $TC_MDP1[1,PLACE]=0
N1000 endfor
```

```
N1010 M30                                          ; End
```

### Display in operating software

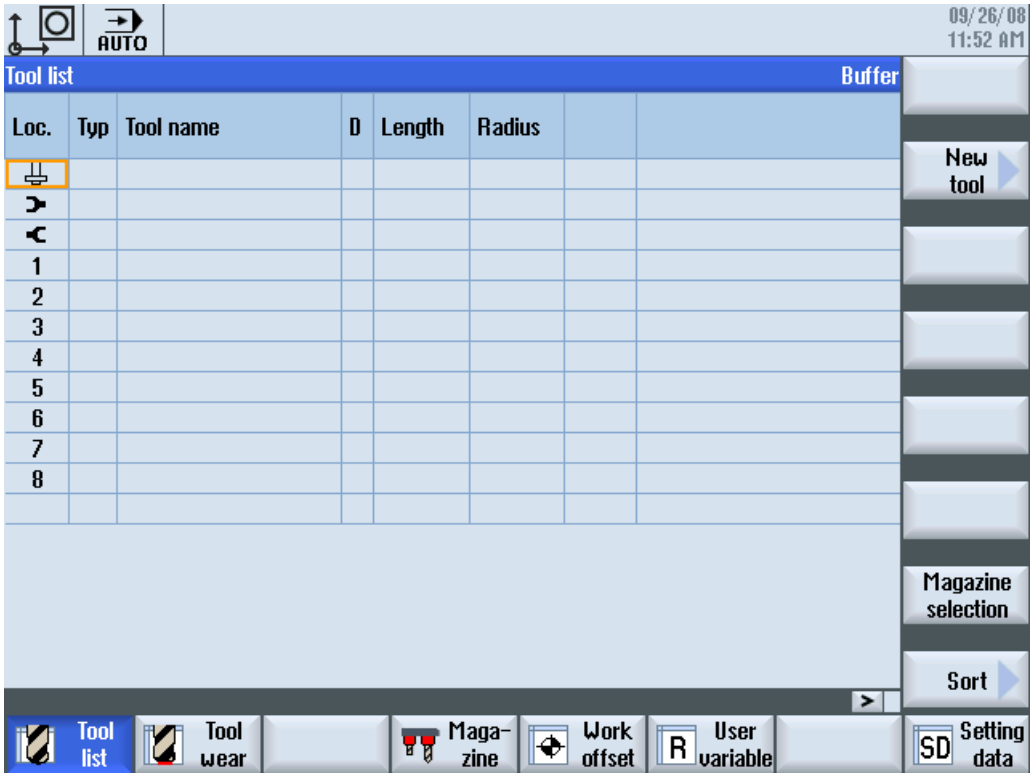



Figure 9-7 Tool list Milling machine

## 9.8.2 Flow chart: Tool change

### Tool change program sequence (PLC)

The sequence described here describes the change between magazine and spindle. The changing of manual tools as well as loading and unloading are not taken into account. These sequences are described in:

- Example: Loading/unloading (Page 300)
- Example: Change manual tools (Page 302)

The machine data default setting is selected such that the job for "Prepare tool change" is triggered with the T command on the interface:

```
N10 T = "Tool name" M6
```

The block preprocessing is not interrupted. M6 is used to start the tool change subprogram (L6) at the same time. As soon as the job for "Prepare tool change" has been acknowledged and in the tool change subprogram the M code for tool change output has been reached, the "Execute tool change" job is output to the interface (block splitting).

A tool change command (M206) must always precede a prepare tool change command. A change command without a previous "Prepare tool change" job will not initiate a job from the tool management.

You will find the expression 1:1 change in the program sequence. This means that the tool change will be carried out in a single cycle. The tool from the spindle (old tool) will be set down at the magazine location of the new tool. No additional magazine positioning is required. In this case, the target location of the old tool in the tool management job is the same as the source location of the new tool (DB43xx.DBW6 and DBW8 are equal to DBW18 and DBW20).

A 1:1 change is not possible for:

- Tools with different location types
- Different tool sizes
- Fixed-location tools

By programming T0 in the NC program, a tool change is initiated without a new tool. Only the spindle tool is transported to the magazine (empty the spindle).

① ... ⑭ refer to the steps in the flow diagram below.

You can see the different acknowledgment options from the steps. Job-related acknowledgments and asynchronous messages are used. Please take the transfer steps used from the table in Section Example: Acknowledgment steps (milling machine) (Page 333)

## "Tool change" flow diagram

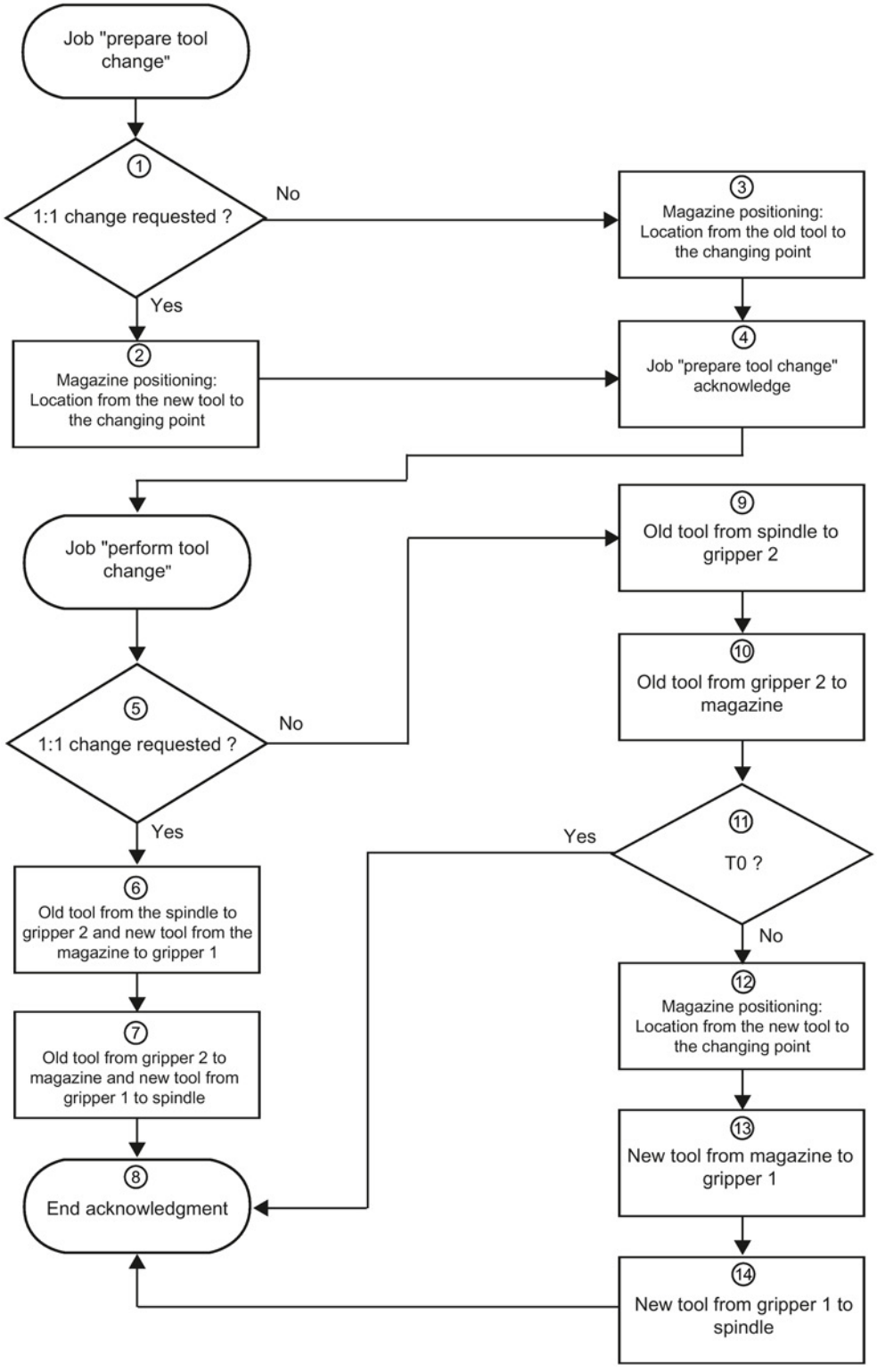



Figure 9-8 Flow diagram

Description of the sequence:

- NC program:

  T command or T command with simultaneous tool change call (M6)

- Interface signals:

  Job from TOOLMAN: DB43xx.DBX0.0 (job bit) and

  DB43xx.DBB1 (command bits): Prepare change (DB43xx.DBX1.2)

- Magazine number and location number of the tools to be moved:

  DB43xx.DBW6 to DBW20: Source location of the new tool, target location of the old tool

### Step 1: 1:1 change requested

Request: Source location of the new tool == target location of the old tool

- Normal situation: There is a tool in the tool holder and a new tool is requested. Both tools have the same location type and the same tool size in the magazine list, the tools are not fixed-location coded.

  The old tool is set down on the magazine location of the new tool, in a direct exchange (1:1 change). If there is no old tool in the tool holder (DB4300.DBX1.4), the same sequence takes place. In this case, the magazine positions the new tool to the changing point.

  **→ Continue with Step 2**

Request: Source location of new tool >< Target location of old tool, Special cases:

- The tool management is informed by DB43xx.DBX1.3 whether the active tool from the tool holder should be removed (T0). In this case, the magazine positions the storage location of the old tool (that is still in the spindle) to the changing point.

  The magazine location for the old tool is also positioned to the changing point if the old tool cannot be stored on the new tool location (1:1 change not possible). The cause could be different location types or tool sizes, or that a fixed-location coded tool is involved. In this case, the tool change is carried out in two steps. First the old tool in the magazine is set down and then the new tool is transported to the spindle.

  **→ Continue with Step 3**

### Step 2: Position magazine, location from new tool to changing point

**← Previous step: Step 1**

| |
|---|
| For example, the magazine movement can be executed via an NC rotary axis controlled by a PLC. The movement should be notified to the tool management. The magazine position is thus updated at the user interface, in the views of the tool and magazine lists. |
| Acknowledgment to TOOLMAN: |
| The target position is read from DB4300.DBW8 (location number for new tool – source) and written to DB9901.DBW2 (transfer step 101). If there is magazine coincidence at the target position, the step is acknowledged asynchronously: |

| Acknowledgment step | Acknowledgment bit | Transfer step new tool | Transfer step old tool | Status |
|---|---|---|---|---|
| 4 | DB4200.DBX0.4 | 101 | 0 | 204 |
| The variable location in magazine 1 is positioned at the changing point to the spindle. | | | | |
| **Transfer step** | **From** | | **To** | |
| | **Magazine** | **Location** | **Magazine** | **Location** |
| 101<br>DB9901.DBW0 | 1 | n | 9998 | 1 |
| n: is the actual location number (n ≠ 0) to be entered by the PLC user program in the variable transfer table. | | | | |

**→ Continue with Step 4**

## Step 3: Position magazine, location from old tool to changing point

**← Previous step: Step 1**

| Same as Step 2, however, the magazine target location is obtained from DB4300.DBW20 (location number for old tool - target).<br>Acknowledgment to TOOLMAN: | | | | |
|---|---|---|---|---|
| **Acknowledgment step** | **Acknowledgment bit** | **Transfer step new tool** | **Transfer step old tool** | **Status** |
| 4 | DB4200.DBX0.4 | 101 | 0 | 204 |
| The variable location in magazine 1 is positioned at the changing point to the spindle. | | | | |
| **Transfer step** | **From** | | **To** | |
| | **Magazine** | **Location** | **Magazine** | **Location** |
| 101<br>DB9901.DBW0 | 1 | n | 9998 | 1 |

n: is the actual location number (n ≠ 0) to be entered by the PLC user program in the variable transfer table.

**→ Continue with Step 4**

### Step 4: Prepare and acknowledge tool change job

**← Previous step: Step 2 or Step 3**

Acknowledgment to TOOLMAN:

- NC program:

  M206 initiates the job for execute tool change
- Interface signals:

  Job from TOOLMAN: DB43xx.DBX0.0 (job bit)

  DB43xx.DBB1 (command bits): Execute change (DB43xx.DBX1.1)
- Magazine number and location number of the tools to be moved:

  DB43xx.DBW6 to DBW20: Source location of the new tool, target location of the old tool

| Acknowledgment step | Acknowledgment bit | Transfer step new tool | Transfer step old tool | Status |
|---|---|---|---|---|
| 1 | DB4200.DBX0.1 | 0 | 0 | 1 |

With this, the preparation for the tool change is complete for many systems.

**→ Continue with Step 5**

### Step 5: 1:1 change requested

**← Previous step: No previous step, entry point in the sequence for "Execute tool change" job**

Similarly to Step 1, it is checked whether a direct change or a change in two steps is to be executed:

- 1:1 change possible: **→ Continue with Step 6**
- 1:1 change not possible: **→ Continue with Step 9**

### Step 6: Old tool from spindle to gripper 2 and new tool from magazine to gripper 1

**← Previous step: Step 5**

The PLC program controls the machine functions with which the gripper movements, tool clamping, etc. are performed. As soon as the mechanical movements are completed and acknowledged in the PLC user program, the tool movements are acknowledged to the tool management.

Acknowledgment to TOOLMAN:

| Acknowledgment step | Acknowledgment bit | Transfer step new tool | Transfer step old tool | Status |
|---|---|---|---|---|
| 5 | DB4200.DBX0.5 | 1 | 2 | 105 |

New tool from magazine to gripper 1 → Step 6 or 13

| Transfer step | From | | To | |
|---|---|---|---|---|
| | Magazine | Location | Magazine | Location |
| 1<br>DB9900.DBW0 | 0 | 1 | 9998 | 2 |

Old tool from spindle to gripper 2 → Step 6 or 9

| Transfer step | From | | To | |
|---|---|---|---|---|
| | Magazine | Location | Magazine | Location |
| 2<br>DB9900.DBW8 | 9998 | 1 | 9998 | 3 |

→ Continue with Step 7

## Step 7: Old tool from gripper 2 to magazine and new tool from gripper 1 to spindle

← Previous step: Step 6

| The PLC program controls the machine functions for the gripper movements, tool clamping, etc. As soon as the mechanical movements are completed and acknowledged in the PLC user program, the tool movements are acknowledged to the tool management.<br>Acknowledgment to TOOLMAN: | | | | |
|---|---|---|---|---|
| **Acknowledgment step** | **Acknowledgment bit** | **Transfer step new tool** | **Transfer step old tool** | **Status** |
| 7 | DB4200.DBX0.7 | 3 | 4 | 105 |

New tool from gripper 1 to spindle → Step 7 or 14

| Transfer step | From | | To | |
|---|---|---|---|---|
| | Magazine | Location | Magazine | Location |
| 3<br>DB9900.DBW16 | 9998 | 2 | 9998 | 1 |

Old tool from gripper 2 to magazine → Step 7 or 10

| Transfer step | From | | To | |
|---|---|---|---|---|
| | Magazine | Location | Magazine | Location |
| 4<br>DB9900.DBW24 | 9998 | 3 | 0 | 2 |

→ Continue with Step 8

## Step 8: End acknowledgment

← Previous step: Step 7 or Step 14

| End acknowledgment takes place with tool changeover in the initial setting, or in a state where the machine can continue machining. Here it is possible that there are still mechanical movements to be executed before the tool change can be completed.<br>Acknowledgment to TOOLMAN: | | | | |
|---|---|---|---|---|
| **Acknowledgment step** | **Acknowledgment bit** | **Transfer step new tool** | **Transfer step old tool** | **Status** |
| 1 | DB4200.DBX0.1 | 0 | 0 | 1 |

→ Continue with Step 9

## Step 9: Old tool from spindle to gripper 2

**← Previous step: Step 5**

| The PLC program controls the machine functions with which the gripper movements, tool clamping, etc. are performed. As soon as the mechanical movements are completed and acknowledged in the PLC user program, the tool movements are acknowledged to the tool management.<br>Acknowledgment to TOOLMAN: | | | | |
|---|---|---|---|---|
| **Acknowledgment step** | **Acknowledgment bit** | **Transfer step new tool** | **Transfer step old tool** | **Status** |
| 8 | DB4200.DBX1.0 | 0 | 2 | 105 |
| Old tool from spindle to gripper 2 → Step 6 or 9 | | | | |
| **Transfer step** | **From** | | **To** | |
| | **Magazine** | **Location** | **Magazine** | **Location** |
| 2<br>DB9900.DBW8 | 9998 | 1 | 9998 | 3 |

**→ Continue with Step 10**

## Step 10: Old tool from gripper 2 to magazine

**← Previous step: Step 9**

| The PLC program controls the machine functions with which the gripper movements, tool clamping, etc. are performed. As soon as the mechanical movements are completed and acknowledged in the PLC user program, the tool movements are acknowledged to the tool management.<br>Acknowledgment to TOOLMAN: | | | | |
|---|---|---|---|---|
| **Acknowledgment step** | **Acknowledgment bit** | **Transfer step new tool** | **Transfer step old tool** | **Status** |
| 9 | DB4200.DBX1.1 | 0 | 4 | 105 |
| Old tool from gripper 2 to magazine → Step 7 or 10 | | | | |
| **Transfer step** | **From** | | **To** | |
| | **Magazine** | **Location** | **Magazine** | **Location** |
| 4<br>DB9900.DBW24 | 9998 | 3 | 0 | 2 |

**→ Continue with Step 11**

## Step 11: T0 ?

**← Previous step: Step 10**

Request: Is T0 set in the tool change job? → DB43xx.DBX1.3

- If only the tool holder should be emptied, the tool change can be completed. → **Continue with Step 8**
- Do you want to place a new tool in the tool holder? → **Continue with Step 12**

## Step 12: Position magazine, location from new tool to changing point

**← Previous step: Step 11**

| Sequence as in Step 2<br>Acknowledgment to TOOLMAN: | | | | |
|---|---|---|---|---|
| **Acknowledgment step** | **Acknowledgment bit** | **Transfer step new tool** | **Transfer step old tool** | **Status** |
| 4 | DB4200.DBX0.4 | 101 | 0 | 204 |
| The variable location in magazine 1 is positioned at the changing point to the spindle. | | | | |
| **Transfer step** | **From** | | **To** | |
| | **Magazine** | **Location** | **Magazine** | **Location** |
| 101<br>DB9901.DBW0 | 1 | n | 9998 | 1 |

n: is the actual location number (n ≠ 0) to be entered by the PLC user program in the variable transfer table.

**→ Continue with Step 13**

## Step 13: New tool from magazine to gripper 1

**← Previous step: Step 12**

| The PLC user program controls the machine functions used to execute gripper movements, tool clamping, etc.<br>Acknowledgment to TOOLMAN: | | | | |
|---|---|---|---|---|
| **Acknowledgment step** | **Acknowledgment bit** | **Transfer step new tool** | **Transfer step old tool** | **Status** |
| 10 | DB4200.DBX1.2 | 1 | 0 | 105 |
| New tool from magazine to gripper 1 → Step 6 or 13 | | | | |
| **Transfer step** | **From** | | **To** | |
| | **Magazine** | **Location** | **Magazine** | **Location** |
| 1<br>DB9900.DBW0 | 0 | 1 | 9998 | 2 |

**→ Continue with Step 14**

## Step 14: New tool from gripper 1 to spindle

**← Previous step: Step 13**

| The PLC user program controls the machine functions with which the gripper movements, tool clamping, etc. are performed. As soon as the mechanical movements are completed and acknowledged in the PLC user program, the tool movements are acknowledged to the tool management.<br>The tool change can be ended:<br>Acknowledgment to TOOLMAN: |
|---|

| Acknowledgment step | Acknowledgment bit | Transfer step new tool | Transfer step old tool | Status |
|---|---|---|---|---|
| 11 | DB4200.DBX1.3 | 3 | 0 | 105 |
| New tool from gripper 1 to spindle → Step 7 or 14 | | | | |
| Transfer step | From | | To | |
| | Magazine | Location | Magazine | Location |
| 3<br>DB9900.DBW16 | 9998 | 2 | 9998 | 1 |

**→ Continue with Step 8**

## 9.8.3 Example: Acknowledgment steps (milling machine)

### Constant transfer step table

| Transfer step | of | | to | | Comment |
|---|---|---|---|---|---|
| | Magazine | Location | Magazine | Location | |
| 1<br>DB9900.DBW0 | 0 | 1 | 9998 | 2 | New tool from magazine to gripper 1<br>Step ⑥ or ⑬ |
| 2<br>DB9900.DBW8 | 9998 | | 9998 | 3 | Step ⑥ or ⑨ |
| 3<br>DB9900.DBW16 | 9998 | | 9998 | 1 | Step ⑦ or ⑩ |
| 4<br>DB9900.DBW24 | 9998 | 3 | 0 | 2 | Step ⑦ or ⑭ |
| 5<br>DB9900.DBW32 | 0 | 2 | 9998 | 1 | Storage location of the old tool |
| 6<br>DB9900.DBW40 | 0 | 1 | 9998 | 1 | Magazine location with the new tool to the changing point<br>Step ② or ⑫ |
| 7<br>DB9900.DBW48 | -- | -- | -- | -- | |

## Variable transfer step table

| Transfer step | of | | to | | Comment |
|---|---|---|---|---|---|
| | Magazine | Location | Magazine | Location | |
| 101<br>DB9901.DBW0 | 1 | n | 9998 | 1 | The variable location in Magazine 1 is positioned at the changing point to the spindle. |
| 102<br>DB9901.DBW8 | -- | -- | -- | -- | |

n : here is the actual location number (n ≠ 0) to be entered by the PLC user program in the variable transfer table.

## Acknowledgment step table

| Acknowledgment step | Transfer step | | Acknowledgment status | Comment |
|---|---|---|---|---|
| | Old tool | New tool | | |
| 1<br>DB9902.DBW0 | 0 | 0 | 1 | End acknowledgment,<br>Step ④ and ⑧ |
| 2<br>DB9902.DBW4 | 0 | 0 | 3 | Cancel order |
| 3<br>DB9902.DBW8 | 0 | 0 | 105 | Intermediate acknowledgment for subsequent order, Step ④ |
| 4<br>DB9902.DBW12 | 101 | 0 | 204 | Variable magazine location to changing point |
| 5<br>DB9902.DBW16 | 1 | 2 | 105 | Intermediate acknowledgment Step ⑥ |
| 6<br>DB9902.DBW20 | 0 | 5 | 105 | Intermediate acknowledgment Step ③ |
| 7<br>DB9902.DBW24 | 3 | 4 | 105 | Intermediate acknowledgment Step ⑦ |
| 8<br>DB9902.DBW28 | 0 | 2 | 105 | Intermediate acknowledgment Step ⑨ |
| 9<br>DB9902.DBW32 | 0 | 4 | 105 | Intermediate acknowledgment Step ⑩ |
| 10<br>DB9902.DBW36 | 1 | 0 | 105 | Intermediate acknowledgment Step ⑬ |
| 11<br>DB9902.DBW40 | 3 | 0 | 105 | Intermediate acknowledgment Step ⑭ |
| 12<br>DB9902.DBW44 | -- | -- | -- | |

**Note:** The step numbers ① ... ⑭ refer to the flow chart in Chapter Flow chart: Tool change (Page 324)

## 9.8.4 Example: Tool change cycle for milling machine

### Sample program

```
PROC L6 SAVE DISPLOF
;---------------------------------------------------------------
; Example of a tool change cycle for machine manufacturers
;---------------------------------------------------------------
DEF INT _WZ_IN_SP,_WZ_VOR
DEF REAL _SPP= ...                        ; spindle position
;
  IF(NOT $P_SEARCH)                       ; if no block search
    _WZ_IN_SP=$TC_MPP6[9998,1]            ; tool in spindle
    GETSELT(_WZ_VOR)                      ; previously selected tool
;
    IF(_WZ_IN_SP<>_WZ_VOR)                ; if another tool
      SPOS=_SPP                           ; position spindle
      G0                                  ; approach tool change position:
      G75 Z=0
      WAITS(1)
    ENDIF
  ELSE
  ENDIF
;
; Load tool: Tool management and PLC
M206
M17
;---------------------------------------------------------------
; END
;---------------------------------------------------------------
```

# Easy Archive 10

## 10.1 Backing up and archiving data

### Overview

In line with the data segregation in data classes, an archive can be created separately for each data area and for each data class.

The data class "System" is an exception. This data is set permanently and becomes effective at the first installation or a default initialization. For this reason, data backup of system data is unnecessary, as this data class contains no data created during commissioning or during the machine run-time.

| NOTICE |
| --- |
| **Protection of the system data**<br>All system data and data contained in the "System" data class in the HMI, NC, PLC and drive areas may not be affected.<br>System data cannot be changed through operation, by writing from a part program, subprogram or cycle and by importing an archive.<br>When you load user data with a USB FlashDrive, the data volume must not be larger than 4 MB! |

### Identification of archives

Every archive contains the following identifiers:

- Data class: M, I, U
- Type of control: For example 828D ME
- Time stamp: Date and time when the archive was created
- Version designation: Software version with which this archive was created
- Serial number of the system CompactFlash card

These markings allow filters to be set when reading in, determining which archive may be read in to which control variant, depending on the data class and the software version.

### See also

Introduction and use of data classes (Page 25)

## 10.2 Which data is backed up in the archive?

### When should you back up the start-up data?

The following times are recommended for performing a data backup:

- After commissioning
- After changing machine-specific settings
- After replacing a hardware component
- Before a software upgrade

### Backing up and restoring data

To back up and restore data, in the "Start-up" operating area press the:

- "Save data" softkey for an internal data backup of the entire memory
- "Commissioning archives" softkey to create a commissioning archive or read in a commissioning archive

| NOTICE |
| --- |
| **USB FlashDrives** |
| USB flash drives are not suitable as persistent memory media. |

### Data areas

The following data areas are backed up in a commissioning archive:

| Components | Data |
|---|---|
| NC data | • Compile cycles<br>• Standard and user cycles<br>• Definitions and macros<br>• Machine data<br>• Setting data<br>• Option data<br>• Global (GUD) and local (LUD) user data<br>• Tool and magazine data<br>• Protection zone data<br>• R parameters<br>• Work offsets<br>• Compensation data<br>• Part programs<br>• Subprograms<br>• Workpieces |
| PLC data | • Main program (MAIN)<br>• Data blocks (DBs)<br>• User program |
| Drive data | Binary format or ASCII format |
| HMI data | • Texts: PLC alarm texts, cycle alarm texts, part program message texts from the machine manufacturer<br>• Templates: Individual templates, workpiece templates<br>• Applications: Software applications, for example of the machine manufacturer<br>• Configurations<br>• Configuration: Configurations, incl. display machine data<br>• Help: Online help files<br>• Version data<br>• Reports: For example, action log, screenshots<br>• Program lists<br>• Dictionaries: For simplified Chinese and traditional Chinese (IME)<br>• Data backups: Channel data, axis data, etc. in ASCII format<br>• Programs on a local drive: Programs that are in the user memory area of the CompactFlash card. |

**Note**

**Archive data**

The commissioning archives are saved taking into account the data classes (*.ard). The drive data is saved as binary data which cannot be edited.

**Compatibility of the PLC data**

SINUMERIK 802D sl PLC archives can be transferred to a SINUMERIK 828D control variant using the Programming Tool.

## Memory areas for archives

Archives can be stored in the following memory areas:

- On the user CompactFlash card: in any directory
- On the CompactFlash card at:

  `/user/sinumerik/data/archive`

  or:

  `/oem/sinumerik/data/archive`
- On a USB FlashDrive

# 10.3 This is how you back up just the machine data that have changed

## Requirement

The access level "Service" is required.

## Delta backup only for machine data that has changed

Using the general MD11210 $MN_UPLOAD_MD_CHANGES_ONLY, you can set whether all machine data should be backed up in the archive, or only the machine data that differs from the default setting:

| MD11210 $MN_UPLOAD_MD_CHANGES_ONLY | |
|---|---|
| Backing up just machine data that has changed | |
| = FFH (default setting) | |
| Bit 0 | |
| . . . | |
| Bit 7 | |

**Note**

**Effect on commissioning archives**

MD11210 $MN_UPLOAD_MD_CHANGES_ONLY is only effective for type *.arc archives.

## Generating an archive

Procedure:

1. Insert the storage medium (CompactFlash card or USB FlashDrive) in the appropriate slot at the front of the control.
2. To generate a type *.arc archive press the keys <Ctrl> + <Alt> + C.

**Result:**

An archive is generated on the storage medium:

The archive name has the following structure: CompleteArchive_Date_Time.arc

**Example:**

CompleteArchive2010-08-11_08-36-15.arc stands for an archive generated on 11.08.2010 at 8:36:15.

# 10.4 This is how you create a commissioning archive

## Overview

Control components can be saved individually or jointly. Creating a separate commissioning archive for each component is recommended, so that the files can be read in again independently of each other.

## Precondition

The access level "Service" is required.

## Creating a commissioning archive

Procedure:

1. Select the "Startup" operating area.
2. Press the menu forward key and then the "Startup archive" softkey.

   The "Startup" window is opened.
3. Select the "Create startup archive" option and confirm with "OK".

   The "Create startup archive" window is opened:

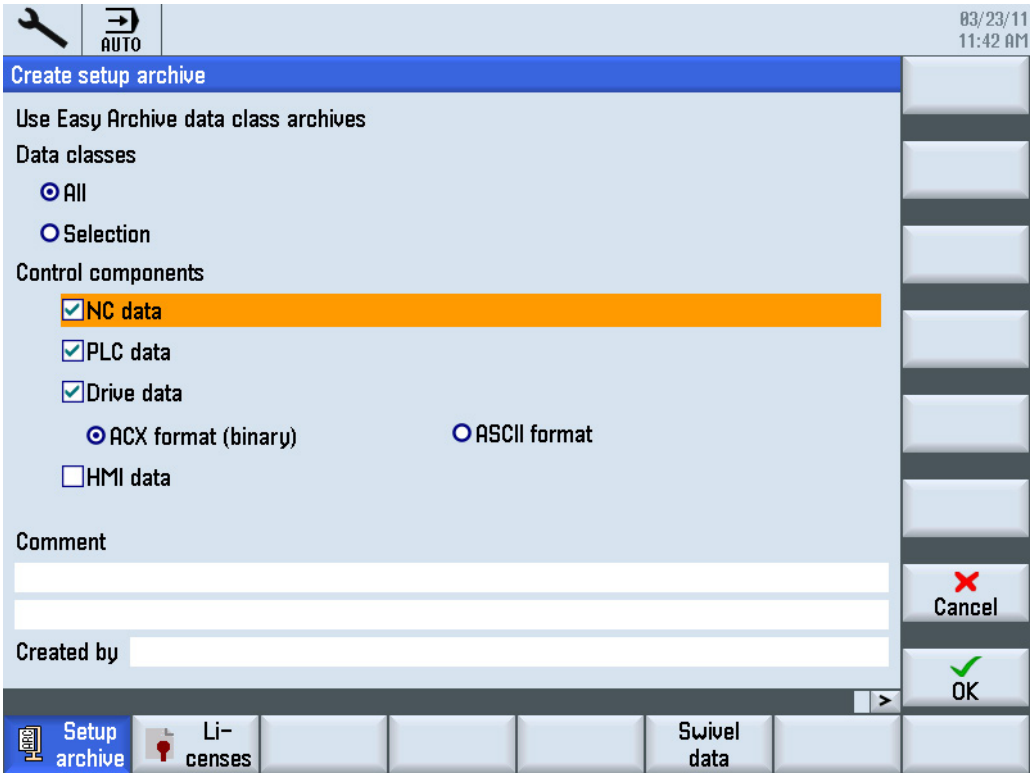



Figure 10-1 Creating a commissioning archive

4. Choose whether the data classes should be "ignored" or "considered" using the <SELECT> key:
   - Select "ignored" to archive all data belonging to the control component.
   - Select "considered" to write only the data in the data classes selected under "Selection" in the archive.
5. Select the control component(s) for the archive.
6. Make use of the option to enter a comment and the creator of the archive.
7. Press the "OK" softkey to create the archive.

   The "Create archive: Select archive" window opens.
8. Select a directory or press the "New directory" softkey to create a new subdirectory.

   The "New Directory" window opens.
9. Enter a name and confirm with "OK."

   The directory is created below the selected folder.

   The "Create archive: Name" window opens.
10. Enter a name and confirm with "OK."

An archive file is created in the selected directory.

## 10.5 This is how you import a commissioning archive

### Precondition

The access level "User" is necessary for reading in an archive.

### Reading-in a commissioning archive

Procedure:

1. Select the "Startup" operating area.
2. Press the menu forward key and then the "Startup archive" softkey.

   The "Startup" window is opened.
3. Select the "Read in startup archive" option and confirm with "OK".

   The "Select startup  archive" window is opened.
4. Select the archive and confirm with "OK".
5. To read in the archive, acknowledge the confirmation prompt with "OK".

   The "Read In Archive" window opens and a progress message box appears for the read-in process.
6. Press the "Cancel" softkey to cancel the read-in process.

# 10.6 Example: Data archiving "Easy Archive" (use case)

## Easy Archive

With "Easy Archive", the SINUMERIK 828D has a fundamentally new procedure for data archiving. This procedure is tailored exactly to the needs of series machine manufacture. "Easy Archive" is based on a strict separation between SINUMERIK system software, adaptation data of the OEM data (machine data, manufacturer cycles) and operator data (part programs, tool offsets). There is a further separation of the adaptation data, into data that is the same for all machines of a particular type, and data that has been adjusted for a specific machine.

This will be clarified with an example:

## User example

A machine manufacturer builds a vertical machining center in series. The adaptation data is created on a prototype machine. This prototype machine's adapted data set will later be ported to all series machines (cloned). After porting the data, however, individual settings are made on each individual machine.

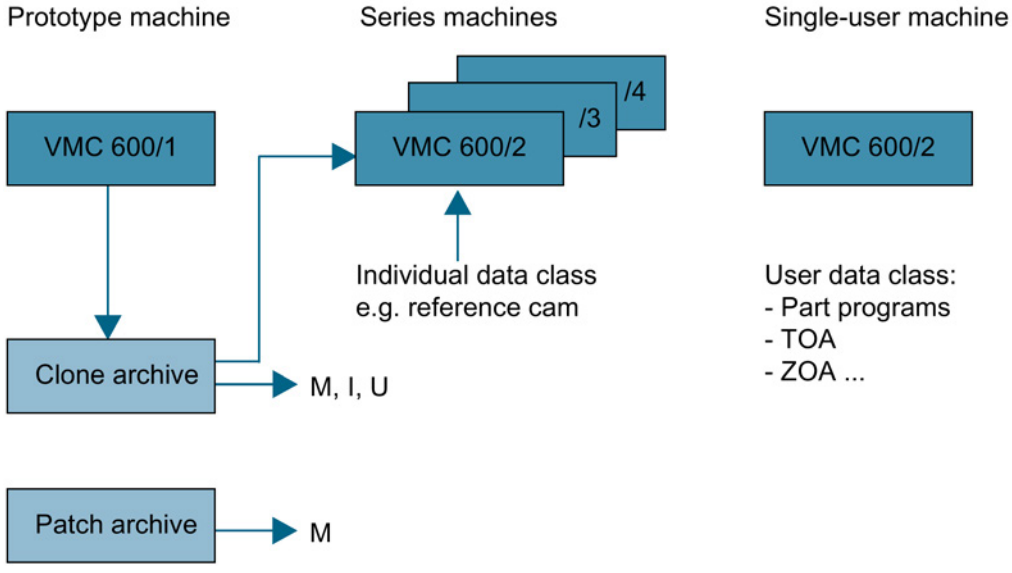



Figure 10-2 User example

For example, reference cams and ball screws are measured and entered as individual adapted data. If an error occurs at the end user, the error is reproduced and solved on the prototype machine. If a complete archive from the prototype machine is now transferred to the affected machine, the individual adapted data of this machine is overwritten by the individual adapted data of the prototype machine.

For the SINUMERIK 828D, the machine manufacturer's adapted data, to which no individual change has been made, may be archived separately. If this archive is transferred to the affected machine, the individual adapted data and the end user data will definitely be retained. So the machine manufacturer's update process is simplified considerably.

## Advantages

The advantage of the "Easy Archive" is that archive creation occurs directly at the SINUMERIK 828D user interface. So no separate PC is needed for the archive.

By separating the system data from the adapted and user data, the SINUMERIK 828D system updates can be completely carried out by the OEMs, without changes to the adapted data. A system update can be carried out by the end users themselves in a short time.

# 10.7 Parameterizing the serial interface

## Data exchange

Data exchange via the serial interface is possible from the following operating areas:

- "Program manager" operating area
- "Start-up" operating area → "System data" softkey

Press the following softkeys to set the interface parameters:




**Note**

If the interface is already assigned, e.g. because a modem is connected, data exchange via the serial interface is not possible and a message is output.

## Activating a GSM modem

You activate a GSM modem using the following machine data:

- MD51233 $MNS_ENABLE_GSM_MODEM = 0 (Voreinstellung)

  The V24 interface is enabled for data transfer.

- MD51233 $MNS_ENABLE_GSM_MODEM = 1

  The GSM modem is activated using this setting. The softkeys for data transfer are no longer displayed.

# Appendix

# A

## A.1 Table: Language code in file names

### Supported languages

System languages:

| Language | Abbreviation in file name |
|---|---|
| Chinese (simplified) | chs |
| Chinese (traditional) | cht |
| German | deu |
| English | eng |
| Spanish | esp |
| French | fra |
| Italian | ita |
| Korean | kor |
| Portuguese (Brazil) | ptb |

Additional languages:

| Language | Abbreviation in file name |
|---|---|
| Czech | csy |
| Danish | dan |
| Finnish | fin |
| Hungarian | hun |
| Indonesian | ind |
| Japanese | jpn |
| Malaysian | msl |
| Dutch | nld |
| Polish | plk |
| Rumanian | rom |
| Russian | rus |
| Slovak | sky |
| Slovene | slv |
| Swedish | sve |
| Thai | tha |
| Turkish | trk |
| Vietnamese | vit |

# A.2 List of abbreviations

| Abbreviation | Meaning | Explanation |
|---|---|---|
| ALM | Active Line Module | |
| ASCII | American Standard Code for Information Inter-change | American coding standard for the exchange of information |
| AUTO | "Automatic" operating mode | |
| BAG | Mode group | |
| OPI | Operator Panel Interface | |
| BERO | Proximity limit switch with feedback oscillator | |
| BICO | Binector Connector | Interconnection technology for the drive |
| CEC | Cross Error Compensation | |
| CNC | Computerized Numerical Control | Computerized numerical control |
| dbSI | drive based Safety Integrated | Safety functions integrated in the drive |
| DB | Data Block in the PLC | |
| DBB | Data Block Byte in the PLC | |
| DBW | Data Block Word in the PLC | |
| DBX | Data Block Bit in the PLC | |
| DDE | Dynamic Data Exchange | Dynamic Data Exchange |
| DDS | Drive Data Set | Drive Data Set |
| DIN | Deutsche Industrie Norm | |
| DO | Drive object | Drive Object |
| DRAM | Dynamic Random Access Memory | Dynamic memory block |
| DRF | Differential Resolver Function | Differential resolver function (handwheel) |
| DRY | DRY run | DRY run feedrate |
| EDS | Encoder Data Set | Encoder data set |
| ESR | Extended Stop and Retract | |
| FIFO | First In - First Out | Method of storing and retrieving data in a memory |
| GUD | Global User Data | Global user data |
| HD | Hard Disk | Hard disk |
| Hardware | Hardware | |
| HSC | High-Speed Cutting | |
| IME | Input Method Editor | Entering Asian characters |
| INC | Increment | Increment |
| INI | Initializing Data | Initializing data |
| IGBT | Insulated Gate Bipolar Transistor | |
| IPO | Interpolator | |
| IRT | Isochronous Real Time | Isochronous communication |
| ISO | International Standardization Organisation | International Standards Organization |
| JOG | "Jogging" operating mode | Jogging via the direction keys |
| LEC | Leadscrew Error Compensation | Leadscrew error compensation |
| LED | Light Emitting Diode | Light-emitting diode |
| LUD | Local User Data | Local user data |

| Abbreviation | Meaning | Explanation |
|---|---|---|
| MB | Megabyte | |
| MCP | Machine Control Panel | Machine control panel |
| MD | Machine data | |
| MDA | "Manual Data Automatic" operating mode | Manual input |
| MDS | Motor Data Set | Motor data set |
| MCS | Machine Coordinate System | |
| MLFB | Machine-Readable Product Code | |
| MPF | Main Program File | Main program (NC part program) |
| MAIN | Main program | Main program (OB1, PLC) |
| MPI | Multi Point Interface | Multi-Point Interface |
| NCK | Numerical Control Kernel | Numerical control kernel |
| NCU | Numerical Control Unit | NCK hardware unit |
| WO | Work Offset | |
| OEM | Original Equipment Manufacturer | |
| PCU | Programmable Control Unit | |
| PI | Program Instance | |
| PG | Programming device | |
| PLC | Programmable Logic Control | Programmable logic controller |
| POU | Program organization unit | in the PLC user program |
| PPU | Panel Processing Unit | Panel-based control |
| PZD | Process data for drives | |
| QEC | Quadrant Error Compensation | Quadrant error compensation |
| REF POINT | "Reference point approach" in JOG mode | |
| REPOS | "Repositioning" in JOG mode | |
| RPA | R-Parameter Active | Memory area on the NCK for R parameter numbers |
| RTC | Real Time Clock | Real-time clock |
| SBL | Single Block | Single block |
| SBR | Subroutine | Subroutine (PLC) |
| SD | Setting Data | |
| SDB | System Data Block | |
| SEA | Setting Data Active | Identifier (file type) for setting data |
| SK | Softkey | |
| SLM | Smart Line Module | |
| SPF | Subprogram file | Subprogram (NC) |
| SRAM | Static Random Access Memory | Static memory block |
| SW | Software | |
| TEA | Testing Data Aktive | Identifier for machine data |
| TO | Tool Offset | Tool offset |
| TOA | Tool Offset Active | Identifier (file type) for tool offsets |
| VPM | Voltage Protection Module | |
| VSM | Voltage Sensing Module | |

| Abbreviation | Meaning | Explanation |
|---|---|---|
| WCS | Workpiece Coordinate System | |
| TMMG | Tool Magazine Management | |
| TM | Tool Management | |
| ZOA | Zero Offset Active | Identifier (file type) for work offset data |

## A.3 Information about third-party software used

Copyright 1995 Sun Microsystems, Inc.

Printed in the United States of America.

All Rights Reserved.

This software product (LICENSED PRODUCT), implementing the Object Management Group's "Internet Inter-ORB Protocol", is protected by copyright and is distributed under the following license restricting its use. Portions of LICENSED PRODUCT may be protected by one or more U.S. or foreign patents, or pending applications.

LICENSED PRODUCT is made available for your use provided that you include this license and copyright notice on all media and documentation and the software program in which this product is incorporated in whole or part. You may copy, modify, distribute, or sublicense the LICENCED PRODUCT without charge as part of a product or software program developed by you, so long as you preserve the functionality of interoperating with the Object Management Group's "Internet Inter-ORB Protocol" version one. However, any uses other than the foregoing uses shall require the express written consent of Sun Microsystems, Inc.

The names of Sun Microsystems, Inc. and any of its subsidiaries or affiliates may not be used in advertising or publicity pertaining to distribution of the LICENSED PRODUCT as permitted herein. This license is effective until terminated by Sun for failure to comply with this license. Upon termination, you shall destroy or return all code and documentation for the LICENSED PRODUCT.

LICENSED PRODUCT IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND INCLUDING THE WARRANTIES OF DESIGN, MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE.

LICENSED PRODUCT IS PROVIDED WITH NO SUPPORT AND WITHOUT ANY OBLIGATION ON THE PART OF SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES TO ASSIST IN ITS USE, CORRECTION, MODIFICATION OR ENHANCEMENT. SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES SHALL HAVE NO LIABILITY WITH RESPECT TO THE INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY LICENSED PRODUCT OR ANY PART THEREOF.

IN NO EVENT WILL SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES BE LIABLE FOR ANY LOST REVENUE OR PROFITS OR OTHER SPECIAL, INDIRECT AND CONSEQUENTIAL DAMAGES, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

SunOS, SunSoft, Sun, Solaris, Sun Microsystems and the Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc.

SunSoft, Inc.
2550 Garcia Avenue
Mountain View, California 94043

Copyright (c) 1991 by AT&T.

Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which

is or includes a copy or modification of this software and in all copies of the supporting documentation for such software.
THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN PARTICULAR, NEITHER THE AUTHOR NOR AT&T MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

This product includes software developed by the University of California, Berkeley and its contributors.
QLocale's data is based on Common Locale Data Repository v1.6.1.

**Note**

You can find additional information on the product DVD in the Readme_OSS file about the third-party software used.

# Index


"

"Speed controller" strategy, 180

$

$MC_TOOL_CHANGE_ERROR_MODE (MD22562), 255, 302
$MC_TOOL_CHANGE_MODE (MD22550), 253
$MN_UPLOAD_MD_CHANGES_ONLY (MD11210), 341
$MN_USER_DATA_INT (MD14510), 239
$MNS_ACCESS_RESET_SERV_PLANNER (MD51235), 234

A

Access levels, 33
Access MyMachine /P2P, 11
Acknowledgment
  Access level, 234
  Maintenance task, 225, 234
  Process, 284
  Rules, 295
  Status, 286
  synchronous, 285
  TM, 267
Acknowledgment block, 227
Acknowledgment step, 293
Acknowledgment-step table, 282
Activating a GSM modem, 347
Actual data table, 224
Additional languages, 349
Adjacent location, 254
Advanced Surface, 78
Alarms
  Configuring variables, 49
  Filter events, 51
  Log, 51
  Structure, 48
Archive
  Memory area, 340
Automatic servo tuning, 176
Auxiliary axes
  Activating, 142
  Configuring, 139
Axis strategy, 178

B

Buffer, 253

C

Certificate of License (CoL), 46
Chain magazine, 253
Circular magazine, 253
Circularity test
  Examples of the position setpoint filter, 214
  Optimization example 1, 215
  Optimization example 2, 216
  Optimization example 3, 217
  Perform measurement, 213
  Save graphic, 218
  Save parameters, 218
  Setting parameters, 213
CNC lock function (option), 84
Commissioning archive, 338
  Create, 342
  Read-in, 344
CompactFlash card, 46
Company network, 19
Controlled system, 173
CYCLE832, 77

D

Data area, 338
  Service Planner, 223
Data backup, 338
Data classes, 26
Data set
  Adding, 162
  Drive (DDS), 161
  Encoder (EDS), 161
  Modify, 171
  Motor (MDS), 161
  Removing, 168
DB1800, 225, 226
DB9903, 223
DB9904, 224
Definition files, 27
Deleting a language, 38

Dictionary
Editing, 39
importing, 39
Direct connection, 22
Drive
BICO interconnection, 111
Circuitry, 154
Control type / setpoints, 111
Encoder assignment, 107, 115
Encoder type selection, 116
Parameter, 157
Terminal wiring, 111
Drive unit
Configuration, 101
Topology, 102
DRIVE-CLiQ
Wiring rules, 146

## E

Easy Archive, 345
Easy Extend, 235
Option bits, 239
EE_IFC (DB9905), 237
End acknowledgment, 285, 286
Ethernet interface, 19
Example
Change manual tool (1), 302
Change manual tool (2), 303
Milling machine, 319
Milling part program, 320
Turning machine, 306
Turning part program, 307

## F

Feedback signal, 269
Friction compensation
Example, 96
Torque injection pulse, 93
Velocity injection pulse, 91

## G

GUD, 27

## H

Hardware serial number, 47
High Speed Cutting (HSC), 77

## I

Infeed
Change for the first commissioning, 123
Configuration, 123
Line data, 124
Overview, 103, 129
Saving the configuration, 128
Summary, 127
Terminal wiring, 126
Initial data table, 223
Input Method Editor (IME), 38
Inputs/outputs (dialog), 156
Interconnections (dialog), 158
Intermediate acknowledgment, 285, 286
ISO Dialect, 76

## J

Job status, 280

## L

Language code, 349
License, 43, 47
License key, 43, 47
License number, 47
Loading magazine, 252

## M

Machine data, 73
Effectiveness, 75
Unit, 74
Macros, 27
Magazine, 252, 291
Configuration, 315
Magazine list, 252
Maintenance interval, 234
Manual tools, 255
Manufacturer PLC alarms, 48
MD10717
$MN_T_NO_FCT_CYCLE_NAME, 260
MD11210
$MN_UPLOAD_MD_CHANGES_ONLY, 341
MD14510
$MN_USER_DATA_INT[i], 239
MD22550
TOOL_CHANGE_MODE, 253
MD22562
TOOL_CHANGE_ERROR_MODE, 255, 302

MD51235
ACCESS_RESET_SERV_PLANNER, 234
MD52270
TM_FUNCTION_MASK, 259
Measuring in JOG, 313
Message, asynchronous, 285
Motor data set direct selection, 167
Multitool, 254

## N

NC variables, 296
Network connection, 22

## O

Option, 43, 47

## P

Parameter
Enable operation/disable operation, 111
password
Change, 35
Set, 35
Path interpolation, 173, 187
Path tuning
With Kp reduction, 186
Without Kp reduction, 186
PI service, 298
Pin assignment
Digital inputs/outputs, 152
PLC alarms, 48
PLC firmware, 251
PLC user program, 251
Aligning, 295
Position controller strategy, 181
Prewarning time MD17300, 85
Probe, 159
Process data, 133
Programming Tool, 11

## R

Replacement tools, 249
Replacing a Motor Module, 130
RTC capacitor, 18

## S

Serial interface, 347
Service Planner
Configuration mode, 229
Standard mode, 230
Setting data, 73
Setting date/time, 36
Spindle, analog, 82
STARTER drive/commissioning software, 12
Strategy selection
Path interpolation, 186
Switch language, 37
System languages, 37, 349

## T

TCP/IP, 19
Telegram type, 133
Terminal assignment, 156
Test license, 43
TMMVTL (PI service), 298
Tool call, 313
Tool change, 272, 324, 325
End acknowledgment, 272
Tool list, 252
Tool management, 249
Block splitting, 325
Toolbox, 11
Total acknowledgment, 286
Transfer step, 291
Transfer-step table
Constant, 281
variable, 282
Tuning objective
Conservative/robust responsiveness, 182
Maximum responsiveness, 182
Moderate responsiveness, 182
Tuning strategy, 178

## U

USB FlashDrive, 338
User example, 345
User interface, 262
User views, 75

## V

V24
Deactivate, 347
Interface, 347
Variable
Location state, 297
Location type, 296
T number, 298

## W

Wiring rules DRIVE-CLiQ, 146

## X

X122 digital inputs/outputs, 151
X132 digital inputs/outputs, 151

SIEMENS

# SINUMERIK

# SINUMERIK 828D
# SINUMERIK Operate

Commissioning Manual


General settings 1

Machine and setting data 2

Connecting drives 3

Configuring the network 4

Customizing the "Machine" operating area 5

Simulation and simultaneous recording 6

Configuring cycles 7

Spindle function 8

Tool management 9

Service and diagnostics 10

Appendix A


Valid for:

CNC software Version 4.7 SP1


01/2015
6FC5397-3DP40-5BA2

## Legal information

### Warning notice system

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

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

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

| ⚠ CAUTION |
| --- |
| 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.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

Order number: 6FC5397-3DP40-5BA2
℗ 01/2015 Subject to change

Copyright © Siemens AG 2014 - 2015.
All rights reserved

# Table of contents


**1 General settings....9**

1.1 Access levels....9
1.1.1 Definition of access levels....9
1.1.2 Modifying the access levels password....9
1.1.3 Access levels for programs....10
1.1.4 Access rights for files....13
1.1.5 Access levels for softkeys....13
1.1.6 This is how you define new access levels for softkeys....14
1.2 Set date/time....16
1.3 Changing the language....18
1.4 Configuring Caps Lock....19
1.5 Inserting a user-specific run up screen....19
1.6 Creating screenshots....20

**2 Machine and setting data....21**

2.1 User views....21
2.1.1 Creating a user view....22
2.1.2 Editing the user view....23
2.1.3 Deleting a user view....25
2.2 Plain texts for machine and setting data....25

**3 Connecting drives....29**

3.1 Overview....29
3.2 Setting up drives....29
3.3 Setting EXTCALL....29
3.4 Setting the EES....30
3.5 Creating a global part program memory....31
3.6 Program selection....31
3.6.1 Displaying the softkey....31

**4 Configuring the network....35**

4.1 Displaying the network overview....35
4.2 Settings of the factory network....36
4.3 Saving network settings....37
4.4 Station-related network diagnostics....38
4.4.1 Displaying the PPU network interfaces....38
4.4.2 TCP/IP diagnostics and configuration....39
4.4.3 Ethernet network diagnostics....42
4.4.3.1 Diagnostics of the network and DHCP server....42

4.4.3.2 Accessible nodes....43
**5 Customizing the "Machine" operating area....47**
5.1 Setting the font size of the actual value display....47
5.2 Inserting a user-specific logo....47
5.3 Configuring the display of the G-code groups....48
5.4 Configuring the channel operational message....49
5.5 Deactivating program test....53
5.6 Activating the "Teach In" function....53
5.7 Block search....55
5.7.1 Activating block search mode....55
5.7.2 Accelerated block search for execution from external....56
5.8 Manual machine....56
5.9 Multi-channel support....58
**6 Simulation and simultaneous recording....61**
6.1 Simulation overview....61
6.2 Setting the technology for simulation....63
6.3 Simultaneous recording overview....65
6.4 Clamping a blank....66
**7 Configuring cycles....71**
7.1 Activating turning/milling/drilling/grinding technologies....71
7.2 Technology cycles for drilling....75
7.3 Manufacturer cycles....77
7.3.1 Manufacturer cycles....77
7.3.2 PROG_EVENT.SPF standard cycle....78
7.3.3 Manufacturer cycle for tool change CUST_T and CUST_M6....79
7.3.4 CUST_TECHCYC.SPF manufacturer cycle....80
7.3.5 CUST_MULTICHAN user cycle....83
7.3.6 Commissioning multi-channel machines with programSYNC....83
7.4 Milling....91
7.4.1 General....91
7.4.2 Technology cycles for milling....91
7.4.3 Setting-up ShopMill cycles for milling....92
7.4.4 Cylinder surface transformation (TRACYL)....94
7.4.5 Example: Milling machine with the XYZ-AC axis configuration....95
7.4.6 ShopMill cycles for multiple clamping....99
7.5 Turning....101
7.5.1 General....101
7.5.2 Setting up cycles for turning....101
7.5.3 Setting-up ShopTurn cycles for turning....105
7.5.4 Setting up a counterspindle....107
7.5.5 Setting up the counterspindle under ShopTurn....109
7.5.6 Technology cycles for turning....110

7.5.7 Axis configuration of a lathe....114
7.5.8 Cylinder surface transformation (TRACYL)....115
7.5.9 End face machining (TRANSMIT)....119
7.5.10 Inclined Y axis (TRAANG)....122
7.5.11 X zero point not at the rotation center....125

7.6 grinding....126

7.7 Swiveling....128
7.7.1 Technology cycles for swiveling....128
7.7.2 CYCLE800 checklist for the identification of the machine kinematics....135
7.7.3 Commissioning swivel data (kinematics chain)....135
7.7.4 Examples of machine kinematics for the commissioning of the Swivel function....143
7.7.5 Manufacturer cycle CUST_800.SPF....151
7.7.6 CYCLE996 measure workpiece kinematics....157

7.8 High Speed Cutting (HSC)....159
7.8.1 High speed settings: Configuring CYCLE832....159
7.8.2 Manufacturer cycle CUST_832.SPF....160

7.9 Measuring cycles and measurement functions....161
7.9.1 Measuring cycles and measurement functions, general....161
7.9.2 Manufacturer and user cycle CUST_MEACYC.SPF....164
7.9.3 Measuring in the JOG mode....166
7.9.3.1 Measuring workpieces at the milling machines....168
7.9.3.2 Measuring tools at the milling machines....170
7.9.3.3 Measuring tools at the turning machines....175
7.9.3.4 Measuring without electronic probe in JOG....176
7.9.4 Measuring in the AUTOMATIC mode....176
7.9.4.1 Measuring workpieces, general ....178
7.9.4.2 Measuring workpieces at the milling machines....181
7.9.4.3 Measuring tools at the milling machines....182
7.9.4.4 Measuring workpieces at the turning machines....188
7.9.4.5 Measuring tools at the turning machines....189

7.10 Compare cycles version....190
7.10.1 Display cycles version....190
7.10.2 Specify cycles version....191

7.11 Cycle protection (option)....193
7.11.1 Overview, cycle protection....193

**8 Spindle function....195**

8.1 Spindle control....195

8.2 Analog spindle....197

8.3 Leadscrew error compensation....197

8.4 Spindle diagnostics....198
8.4.1 Spindle diagnostics....198
8.4.2 Temperatures....200
8.4.3 Motor temperature sensor....202
8.4.4 Additional temperature sensor....202
8.4.5 Temperature histograms....203
8.4.6 Speed/torque....203
8.4.7 Clamping system....204

8.4.8 Clamping system: Speed limits....206
8.4.9 Clamping system: Diagnostic statistics....207
8.4.10 Clamping system: Clamping time statistics....208
8.4.11 Fetching the logistics data....208

**9 Tool management....211**

9.1 Machine data for tool management....211
9.1.1 Settings with/without magazine management....211
9.1.2 Configuring the access levels of the tool management....212
9.1.3 Additional settings....214

9.2 Configuring the user interface....217
9.2.1 General settings....219
9.2.2 Configure the tool lists....224
9.2.2.1 Tool parameter identifiers....228
9.2.2.2 Cutting parameter identifiers....230
9.2.2.3 Monitoring parameter identifiers....232
9.2.2.4 Grinding parameter identifiers....233
9.2.2.5 Magazine location parameter identifiers....234
9.2.2.6 Magazine location adapter parameter identifiers....235
9.2.2.7 Multitool parameter identifiers....235
9.2.2.8 Multitool location parameter identifiers....236
9.2.3 Configure the list parameters....237
9.2.4 List of tool types....241
9.2.5 Configure tool types....243
9.2.6 Configuring the "More data" window....244
9.2.7 Configure the "New tool - favorites" window....246
9.2.8 Configuring the "New Tool" window....246
9.2.9 Configuring default values for new tools....247
9.2.10 Configuring the "Details" window....251
9.2.11 Assignment of TO units and toolholders to channels....251
9.2.12 Assigning a name for magazines and magazine locations....254
9.2.13 Assigning a name for magazine location types....256
9.2.14 Assignment of magazines to channels....257
9.2.15 Coolant and tool-specific functions....259
9.2.16 Reason for the tool change when reactivating....262
9.2.17 Configure the code carrier connection....265

9.3 Creating OEM texts....268
9.3.1 Creating OEM texts....268
9.3.2 Identifiers of the standard texts....269
9.3.3 Examples of OEM texts....271

9.4 Examples....273
9.4.1 Example: Configuring an OEM tool list....273
9.4.1.1 Adapting the configuration file....273
9.4.1.2 Adaptation of the customer text file....275
9.4.2 Example: Configuring magazine location types with names....277
9.4.2.1 Adapting the configuration file....277
9.4.2.2 Adaptation of the customer text file....279

9.5 Replace T, M or D function....281

9.6 Working with two toolholders....284

9.7 Deploy multitool....286
9.7.1 Enable the display of a multitool....286
9.7.2 Loading and unloading multitool....287
9.7.3 Tool change for a multitool....289

9.8 Editor for the tool and magazine management....291
9.8.1 Overview....291
9.8.2 Calling the tool management....292
9.8.3 Editing the tool management....293
9.8.4 Parameterizing the tool management....295
9.8.4.1 Create tool unit....295
9.8.4.2 Define buffer locations....296
9.8.4.3 Define loading location....297
9.8.4.4 Create magazine configuration....297
9.8.4.5 Link assignments....299

**10 Service and diagnostics....301**

10.1 NC/PLC variables....301
10.1.1 Displaying and editing PLC and NC variables....301
10.1.2 Saving and loading screen forms....304

10.2 Displaying the Service overview....305
10.2.1 Selecting axes and drives....307
10.2.2 Axis diagnostics....309
10.2.3 Service drive....313

10.3 PROFINET diagnostics....321
10.3.1 Displaying the PROFINET configuration....321
10.3.2 Displaying details for PROFINET devices....323

10.4 Trace....324
10.4.1 General procedure....326
10.4.2 Trace session....326
10.4.2.1 Creating a session file....326
10.4.2.2 Saving the trace file....327
10.4.2.3 Load trace session file....328
10.4.3 Variable for Trace....329
10.4.3.1 Variables filter/search....329
10.4.3.2 Selecting attributes of a variable....330
10.4.3.3 Displaying details of a variable....333
10.4.4 Trace settings....334
10.4.4.1 Trace settings (PLC, NC, servo)....334
10.4.4.2 Trace settings (drive)....334
10.4.4.3 Trace options....336
10.4.4.4 Starting the trace....336
10.4.5 Evaluate a trace....337
10.4.5.1 Setting trace views....337
10.4.5.2 Selecting a variable....338
10.4.5.3 Scaling the display....339
10.4.5.4 Zooming the display....340
10.4.5.5 Position the cursor....341
10.4.5.6 Acquiring measured values....342

10.5 Drive system....343

10.5.1 Displaying drive states....343
10.5.2 Displaying details of the drive objects....344
10.6 Action log....345
10.6.1 Setting the action log....345
10.6.2 Displaying the log file....347
10.6.3 Searching in the log files....348
10.6.4 Storing a log....348
10.6.5 Structure of a log file....349
10.6.6 Advanced settings....351
10.7 HMI trace....352
**A Appendix....355**
A.1 Information about third-party software used....355
A.2 Abbreviations....356
**Index....359**

# General settings

# 1

## 1.1 Access levels

### 1.1.1 Definition of access levels

Access to programs, data and functions is protected in a user-oriented hierarchical system of seven access levels. These are divided into

- Three password levels for manufacturer, service and users
- Four keyswitch positions for users

There are access levels 1 to 7 (see table below); where

- 1 is the highest and
- 7 is the lowest level.

Table 1-1 Access level concept

| Access level | Protected by | Area |
|---|---|---|
| 1 | Password: SUNRISE (default value) | Manufacturer |
| 2 | Password: EVENING (default value) | Service |
| 3 | Password: CUSTOMER (default value) | User |
| 4 | Keyswitch 3 | Programmer, machine setter |
| 5 | Keyswitch 2 | Qualified operator |
| 6 | Keyswitch 1 | Trained operator |
| 7 | Keyswitch 0 | Semi-skilled operator |

### 1.1.2 Modifying the access levels password

#### Overview

You can activate the access levels protected with passwords from the user interface.

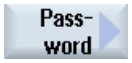
The following options are available for editing the passwords:

- Set new password
- Change password
- Delete password

### Setting the password




1. Select the "Start-up" operating area.




2. Press the "Password" softkey.




3. Press the "Set password" softkey.
   The "Set Password" window opens.




4. Enter one of the possible default passwords and press the "OK" softkey to confirm the entry.
   A permissible password is set and the valid access level is displayed. Invalid passwords will be rejected.

### Change password

You must change the standard passwords to obtain a secure access protection.




1. Press the "Change password" softkey.
   The "Change Password" window opens.
2. Select the area (e.g. user) for which you want to assign a new password.
3. Enter the new password in the input fields "New password" and "Repeat password".




4. Press the "OK" softkey to confirm your entry.
   A new, valid password is only accepted if the two terms entered are identical.

### Delete password




Press the "Delete password" softkey. The access authorization is reset.

**Note**

Access authorization is not automatically deleted at POWER ON.

When a general NC reset is performed, the passwords are reset to the default passwords.

## 1.1.3 Access levels for programs

### Objective

Access levels can be allocated to protect information; they can also be used to prevent unauthorized operation or deliberate damage to the control system or machine.

With access authorization levels to files and directories, it can be defined as to which users are authorized to execute which operations on a directory or a file in a file system. Operations in this sense include:

- Reading the contents of a file (READ), copying the file or placing the file in an archive.
- Changing the contents of a file (WRITE) or deleting the file.
- Executing a file (EXECUTE).
- Listing and displaying a file in a directory (SHOW).
- Creating a file in a directory.
- Displaying or reading the contents of a directory.
- Creating or deleting a directory.

## Description of functions

The **individual allocation of access rights** acts, in the passive NC file system, on the following directories:

- Part programs
- Subprograms
- Manufacturer cycle directory
- User cycle directory
- Workpieces directory
- Workpiece directories

Further, the function acts on the files of the main program type (*.mpf) and subprograms or cycle (*.spf).

In addition, the technique can be expanded to include the following directories on the local drive:

- Part programs
- Subprograms
- Workpieces directory
- Workpiece directories

**Note**

**Working via remote access**

When working via remote access (e.g. WinSCP or RCS Commander) the same access rights apply for editing or saving files and directories as directly at the controller.

## Access rights via machine data

The following machine data is still available to **globally allocate access rights**:
MD51050 $MNS_ACCESS_WRITE_PROGRAM

Default setting = 4 (access level: User)

**Note**

**Effect of access rights**

The more restrictive setting is always effective both when individual access rights have been allocated and when settings have been made via MD51050 $MNS_ACCESS_WRITE_PROGRAM.

## Settings in the slfsfileattributes.ini file

The slfsfileattributes.ini file is available as a template in the following directory:

../siemens/sinumerik/hmi/template/cfg

The values from 0 ... 7 correspond to the access levels that, for example, also apply to the password. The 4-digit numbers designate the sequence according to the following rights:

| Significance from left to right: R W X S | | |
|---|---|---|
| R | READ | Reading files and directories |
| W | WRITE | Changing, deleting or creating files and directories |
| X | EXECUTE | Executes program: For files, for directories, always 7 |
| S | SHOW | Outputs a list of files and directories |

Copy the slfsfileattributes.ini file into one of the following directories:

../**oem**/sinumerik/hmi/cfg or ../**user**/sinumerik/hmi/cfg

The operating software must be restarted in order that the changes become effective.

## Example

The paths are composed of the data type identifiers. In the following example, "\wks.dir*.wpd *.mpf" comprises the data type for workpiece directories "wks.dir" the workpieces "*.wpd" and the part programs "*.mpf".

For all workpieces, 7777 is defined as default setting for new part programs.

All entries are in the section [ACCESSMASKS] and have the following form:

```
[ACCESSMASKS]

\wks.dir\*.wpd\*.mpf = 7777
\mpf.dir\*.mpf = 6577
\cus.dir\*.spf = 6577
```

## See also

Definition of access levels (Page 9)

## 1.1.4 Access rights for files

### Changing access rights

Access rights for execution, writing, listing and reading are displayed in the "Properties" window.

- Execute: is used for the selection for execution.
- Write: controls the changing and deletion of a file or a directory.

For NC files, you have the option to set the access rights from keyswitch 0 to the current access level, to be set separately for each file.

If an access level is higher than the current access level, it cannot be changed.

For external files (e.g. on a local drive), the access rights are displayed to you only if settings have been executed for these files by the machine manufacturer. They cannot be changed via the "Properties" window.

**Note**

**Settings for the access rights to directories and files**

Via the configuration file and MD51050, access rights of the directories and file types of the NC memory and user memory (local drive) can be changed and pre-assigned.

### Configuration file slfsfileattributes.ini

The access rights defined in the configuration file slfsfileattributes.ini and in MD51050 become valid in the NC when creating new files and can then be changed individually in the Program Manager with the "Properties" function. The access rights are an individual attribute of each file in the NC.

The access rights defined in the configuration file slfsfileattributes.ini and MD51050 become valid for external data on the CompactFlash Card of the NCU, or on the hard disk of the PCU/PC immediately for all the specified directories and file types, and cannot be changed through operation. The access rights can only be set globally for external data.

## 1.1.5 Access levels for softkeys

### Use

The display and operation of softkeys can be suppressed by both the OEM as well as the user. This allows the operating software to be specifically adapted to the required functional scope and therefore be configured as transparently as possible.

This means that the functional scope of the system is restricted, to prevent access to functions in the operating software, or to restrict the possibility of operator errors.

### Access hierarchy

A new access level can only be allocated for those softkeys that are also visible and can be used in the actual access level.

Example: If a softkey is only accessible with the "Manufacturer" access level, then the access level cannot be changed by an operator with access level "Key-operated switch setting 3".

### Restrictions

In order to guarantee that the system remains stable and operable, certain softkeys are excluded from the access level change, e.g. the "Customize softkeys" softkey itself, in order that the undo path is not cut-off or the softkey "<< Back" to the vertical softkey bar.

The following softkeys cannot be hidden or the access level changed:

- All "OK" softkeys
- All "Cancel" softkeys
- All "Accept" softkeys
- All "**<<**" Back and "**>>**" Continue softkeys

The access level for softkeys in the cycle dialog boxes can also be changed; this does not apply to softkeys in Easy Screen dialog boxes.

**Note**

**Software upgrade**

As all changes of the access levels for softkeys are described using the softkey ID, they still remain valid after a software upgrade, which may have resulted in softkeys being shifted.

Softkeys, where access is already protected by machine data - for example MD51045 $MNS_ACCESS_TEACH_IN - are only protected by the allocated access level after a new one has been allocated. The machine data is no longer evaluated.

All of the softkeys hidden using this mechanism can be temporarily displayed again using the access rights specified by the system. This is done using the "Show all softkeys" softkey. The softkey is deactivated as long as no softkeys have been allocated new access levels: This setting remains effective until the system is rebooted.

## 1.1.6 This is how you define new access levels for softkeys

### General sequence

In order to assign a new access level to a softkey, proceed as follows:

- Activate the adaptation mode.
- Change the access level of a softkey.
- Accept the changes and confirm.

## Preconditions

Access to the "Customize softkeys" is protected using a general MD51073 $MNS_ACCESS_SET_SOFTKEY_ACCESS. This machine data is pre-assigned access level 3, i.e. the function is available from "Service" password and higher. In order that an operator with "User" access level can use this function, this machine data should be set to a value > 3.

An external mouse is required in order to make changes.

## Changing the access level of a softkey

Procedure:

1. In the "Start-up" operating area, select the "HMI" menu.
2. On the expansion bar "**>>**" select the "Customize softkeys" softkey.
   You can obtain precise information about the procedure in the following dialog.
1. Confirm with "OK", in order to activate the "Customize the softkey access levels active" mode. A table is displayed.
2. Select an operating area.
3. Place the mouse cursor on a softkey and using the right-hand mouse key call the context menu "Define access level", in order to define a new access level for this softkey.
   - The displayed access level is the default setting.
   - A softkey is practically made invisible with "Always hide".
   - You can reset the additionally allocated access level using "Original setting". This selection is only listed after an individual access level was set.
4. In order to accept changes, return to the "Start-up" operating area and there, press the "Close customizing" softkey. This means that you exit the "Customize the softkey access levels active" mode.
   If you acknowledge the following query with "OK", then the changes are accepted and become effective once the system has rebooted.
5. If the changes to a softkey are not to be applied, select the line and press the "Original setting" softkey. This means that in the "New access level" column, the "Original setting" identifier is entered, which functions in the same way as deletion. This entry is no longer available following a system restart. "

## Example

The following access levels were changed:

| Softkey | New access level |
|---|---|
| SISuMainScreenMenuHu::0::3 | Manufacturer |
| SIMaJogMillMenuHU::jogHuMore::6 | User |
| SISuPaUserDataMenu::::4 | Service |
| | |
| | |

Once selected, the identifier and position of the softkey are transferred to the "Softkey" column.

### Result

When the changes have been accepted, a configuration file with the "slsoftkeyaccess.xml" name is created and is saved in the file system under the following path: ../**user**/sinumerik/hmi/cfg. Files can also be saved in the **/oem** path, which are evaluated when the system boots. The following priority applies: oem < user.

**Note**

**Editing the file "slsoftkeyaccess.xml"**

In order to avoid errors in the xml syntax and for the softkey identifier the "slsoftkeyaccess.xml" file may only be edited via the operating software as described above, and not directly in the file system under "System data". It is possible that the complete file or individual entries for softkeys can no longer be interpreted.

## 1.2 Set date/time

You have the possibility of manually setting the date and time for the SINUMERIK control. In so doing, you can select from various display formats.

### Formats of the date display

| Formats | Example |
|---|---|
| d.M.yy | 3.1.09 |
| dd.MM.yy | 03.01.09 |
| dd.MM.yyyy | 03.01.2009 |
| d/M/yy | 3/01/09 |
| dd/MM/yy | 03/01/09 |
| dd/MM/yyyy | 03/01/2009 |
| M/d/yy | 1/3/09 |
| MM/dd/yy | 01/03/09 |
| MM/dd/yyyy | 01/03/2009 |
| yyyy/MM/dd | 2009/01/03 |

The following applies:

| Date | Description of the entry |
|---|---|
| d | Single or double digit day of the month (1 - 31). |
| dd | Double digit day of the month, with leading zero (01 - 31). |
| M | Single or double digit month of the year (1 - 12). |
| MM | Double digit month of the year, with leading zero (01 - 12). |
| yy | Double digit year (00 - 99). |
| yyyy | Four-digit year (e.g. 2009). |

## Time display formats

| Formats | Example | Hour system |
|---|---|---|
| H:m:s AP | 1:2:4 PM | 24 hour system, with display of "AM" / "PM" |
| h:m:s ap | 1:2:8 pm | 12 hour system, with display of "am" / "pm" |
| HH:mm:ss AP | 13:02:08 PM | 24 hour system, with display of "AM" / "PM" |
| hh:mm:ss ap | 01:02:09 pm | 12 hour system, with display of "am" / "pm" |
| H:mm:ss AP | 1:02:09 PM | 24 hour system, with display of "AM" / "PM" |
| h:mm:ss ap | 1:02:09 pm | 12 hour system, with display of "am" / "pm" |
| h:mm:ss | 13:02:09 | 24 hour system |
| hh:mm:ss | 13:12:04 | 24 hour system |

The following applies:

| Time symbol: | Description of the entry | |
|---|---|---|
| h | Single or double digit hour (0 - 23 or 1 - 12 for am/pm). | |
| hh | Double digit hour, with leading zero (00 - 23 or 01 - 12 for am/pm). | |
| H | Single or double digit hour (0 - 23 for AM/PM). | |
| HH | Double digit hour, with leading zero (00 - 23 for AM/PM). | |
| m | Single or double digit minute (0 - 59). | |
| mm | Double digit minute, with leading zero (00 - 59). | |
| s | Single or double digit second (0 - 59). | |
| ss | Double digit second, with leading zero (00 - 59). | |
| AP | Display of the time of day: | AM or am = before midday or<br>PM or pm = after midday |

## Procedure




1. Select the "Start-up" operating area.




2. Press the "HMI" softkey.




3. Press the "Date/Time" softkey.
   The "Set Date and Time" window opens.




4. Select the required formats for the date and time with the <SELECT> key in the "Format" field.

5. In the entry fields after "new" enter the actual date and time.




6. Confirm the entry with the "OK" softkey.
   The new date and time details are transferred.

# 1.3 Changing the language

The operating software is available as standard in several languages.

## Selecting a language

You can define the language selection mode using the following machine data.

| MD9100 $MM_CHANGE_LANGUAGE_MODE | Language selection mode |
|---|---|
| = 1 | The user-interface language is specified via the "Language Selection" window (default value). |




1. Select the "Start-up" operating area.




2. Press the "Change language" softkey.
   The "Language Selection" window opens. The language most recently set is highlighted.
3. Switch the cursor to the required language.




4. Press the <INPUT> key.

   - OR -




   Press the "OK" softkey.
   The operating software switches to the language selected.

## Defining two languages

| MD9100 $MM_CHANGE_LANGUAGE_MODE | Language selection mode |
|---|---|
| = 2 | Selection of two languages. During operation, the "Change language" softkey can be used to toggle between these languages. |




1. Select the "Start-up" operating area.




2. Press the "HMI" softkey.



3. Press the ">>" softkey.




4. Press the "Language selection" softkey.
   The "Language Selection" window opens.
   All the installed languages are displayed in the fields "First language" and "Second language".
5. Select a language in each of these.




6. Press the <INPUT> key.

- OR -




Press the "OK" softkey.




During operation, you can switch between the two languages by pressing the "Change language" softkey.

## 1.4 Configuring Caps Lock

The "CAPSLOCK on" function ensures that text entered via an external keyboard is always entered in upper case and not in lower case.

The key behavior is set using the following display machine data.

### Setting

| MD9009 $MM_KEYBOARD_STATE | |
|---|---|
| = 0 | CAPSLOCK off (default value) |
| = 2 | CAPSLOCK on |

If the machine data is changed, the system must be rebooted.

## 1.5 Inserting a user-specific run up screen

### OEM-specific run up screen

You can replace the Siemens run up screen by your own run up screen.

Procedure:

1. Call your own run up screen "`splash.png`".

   **Note**

   The file name must not contain any lower case letters.

2. Save your own run up screen in one of the following directories:
   /**user**/sinumerik/hmi/ico/ico640 or
   /**oem**/sinumerik/hmi/ico/ico640 or
   /**addon**/sinumerik/hmi/ico/ico640

   **Note**

   The size of the screen is of no significance. A resolution of 640x480 pixels is always set.

# 1.6 Creating screenshots

You can create screenshots of the current user interface.

Each screenshot is saved as a file and stored in the following folder:

/**user**/sinumerik/hmi/log/screenshot

## Procedure

Ctrl + P Press the <Ctrl+P> key combination.

A screenshot of the current user interface is created in .png format.

The file names assigned by the system are in ascending order from "SCR_SAVE_0001.png" to "SCR_SAVE_9999.png". You can create up to 9,999 screenshots.

## Copy file




1. Select the "Start-up" operating area.




2. Press the "System data" softkey.
3. Open the folder specified above, and select the required screenshots.




4. Press the "Copy" softkey.

   - OR -




   Press the "Cut" softkey.




5. Open the required archive directory, e.g. on a USB flash drive and press the "Paste" softkey.

# Machine and setting data

# 2

## 2.1 User views

### Usage

In the "User Views" window you can create and adapt all machine data relevant to a particular functionality. Machine data is individually collated to simplify the user's task.

You can include additional comments when creating and/or editing user views.

### Display in the the "Parameter" operating area

Proceed as follows to show user views also in the "Parameters" operating area:

1. Select the "Start-up" operating area.
2. Press the "Machine data" softkey.
3. Press the "User views" softkey.
4. Press the "Manage view" softkey.
5. Press the "New view" softkey.
6. Activate the "Use display also under parameters / setting data" checkbox if you also wish to display the new user view in the "Parameters" → "Setting data" → "Data lists" operating area.
7. Enter a name for the user view.
8. Confirm with "OK".
9. Insert the machine data that is to be displayed in the "Parameters" operating area.

## 2.1.1 Creating a user view

### Procedure




1. Select the "Start-up" operating area and press the "Mach. data" softkey.







2. Press the softkeys "User views",
"Manage view" and







"New view".
The "New View" window opens.

3. Enter the desired view name.
Press the "OK" softkey.
If a user view already exists with the same name, you receive a safety prompt as to whether the existing view should be overwritten.







4. Press the "Enter data" softkey.
The "Insert data" selection box opens.




5. Open the selection list with the <INSERT> key, select a machine data area and press the <INPUT> key.








Select a machine data item with the cursor keys or using the "Find" softkey.










6. Press the "Insert before line" or "Insert after line" softkey to insert the selected machine data at the required position in the user view.

**Note:**

You can navigate in the open user view with the mouse and select a line without closing the list box.




7. Press the "Back" softkey to close the list box and return to the user view you want to edit.
   Your entries are automatically stored.




8. Press the "Insert text" softkey to insert any text in the open user view.
   The "Insert Text" input window opens.







9. Enter a text and explanatory description for the text and press the "Insert before line" or "Insert after line" softkey to accept the text.




10. Press the "Back" softkey to save your entries and return to the user view you are editing.

### Selecting a particular machine data item




You can use the Search dialog box to look for a particular data item.

## 2.1.2 Editing the user view

Depending on the row you have selected, you can use the "Properties" softkey to change comments and, in the case of machine data, the data source (channel, axis, drive unit) and view the description and, if required, change.

- Machine data
  Depending on the data type you can select either a fixed or a variable data source for machine data.
  The entry "variable(*)" means that when you can select another data source (channel, axis, drive unit) with vertical softkeys 1- 3 (+, - and direct selection) the value displayed changes to refer to the data source currently selected.
  Alternately, you can select a fixed data source so that the displayed value does not change when you select a different data source.
- Comment texts
  Text and description

### Procedure




1. Select the "Start-up" operating area and press the "Mach. data" softkey.



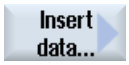




2. Press the "User views" softkey.
   The "User Views" window opens.




3. Press the "Edit view" softkey. Additional softkeys to edit the user view are listed.




4. Press the "Insert data" softkey if you wish to enter another data item.



5. Select a machine data item with the cursor keys.






6. Press the "Up" or "Down" softkeys to move the selected row.




- OR -




Press the "Delete line" softkey to remove the selected line from the view. The data item is removed without a prompt.

- OR -




Press the "Properties" softkey to view the texts for comments and, if required, change.

The "Properties" window is opened. For machine data, depending on the setting, the description or the machine data is displayed in the lower left-hand window.




7. Press the "Back" softkey to save your changes.

## Selecting a particular machine data item




You can use the Search dialog box to look for a particular data item.




Press the "Go to start" softkey to start the search at the first entry.




Press the "Go to end" softkey to start the search at the last entry.

Press the "Continue search" softkey if the data found during the search does not match up with what you are looking for.

### 2.1.3 Deleting a user view

#### Procedure




1. Select the "Start-up" operating area.




2. press the "Machine data" softkey.




3. Press the "User views" softkey.




4. Press the "Manage view" softkey.

5. Select the desired view from the selection list.




6. Press the "Delete view" softkey.




With the final confirmation prompt you will be requested to confirm with either "Yes" or cancel with "No".

or




The "User views" dialog is then redisplayed.

Just like before, as an alternative, it is possible to delete a user view under "System data" under the following path: `../user/sinumerik/hmi/template/user_views`

## 2.2 Plain texts for machine and setting data

Application-specific and language-dependent plain texts can be configured for machine and setting data depending on the version [Index].

#### Displaying the plain texts

You set the display of the plain texts using the following display machine data:

| MD9900 $MM_MD_TEXT_SWITCH | Plain texts instead of MD identifier |
|---|---|
| = 1 | Plain text is displayed in the table |
| = 0 | Plain text is displayed in the dialog line |

## Files

| File | Machine/setting data |
|---|---|
| nctea_xxx.txt | General machine/setting data |
| chtea_xxx.txt | Channel-specific machine/setting data |
| axtea_xxx.txt | Axis-specific machine/setting data |
| hmitea_xxx.txt | Display machine data |

The specified language code must be used for "xxx", e.g. "deu" for German, "eng" for English, etc.

Ensure that the file names are written in lower case letters.

## Procedure

1. Depending on the machine or setting data for which you wish to change the plain text, create the corresponding file, e.g. "nctea_xxx.txt".
2. Save the file with the UTF-8 coding in order that umlauts and special characters are correctly displayed in SINUMERIK Operate.
3. Place the file in SINUMERIK Operate in the following directory:
   System CF card/**user**/sinumerik/hmi/lng or
   System CF card/**oem**/sinumerik/hmi/lng or
   System CF card/**addon**/sinumerik/hmi/lng

   **Note**

   The files are searched through in the sequence of the directories "user", "oem" and "addon". The first file that is found is evaluated. If, e.g. text files are available in the "user" and "oem" directories, the files are evaluated in the "user" directory.

**Note**

If a machine data item has different indices and you do not specify an index for the plain text, the same plain text appears for all the indices of the machine data item.

The parameters and indices must be sorted in an ascending order.

## Example of a file with German texts

```
14510 Mein Integer-Text für PLC_MD 14510
14510[1] Mein Integer-Text für MD 14510[1]
14512[1] Mein Text für HEX-Datum 14512[1]
```

## Result in the display

The plain text is displayed in the table or in the diagnostics line in the operating area "Startup", "General MD".

| Machine data | Text |
|---|---|
| 14510 | My integer text for PLC_MD 14510 |
| 14510[1] | My integer text for MD 14510[1] |
| 14512[1] | My text in HEX data 14512[1] |

# Connecting drives 3

## 3.1 Overview

Up to 21 connections to so-called logical drives (data carriers) can be configured. These drives can be accessed in the "Program manager" and "Startup" operating areas.

The following logical drives can be set up:

- USB interface
- CompactFlash card
- Network drives



**Software option**

You will need the "Manage network drives" option to manage additional drives via Ethernet.

## 3.2 Setting up drives

The "Setup Drives" window is available in the "Start-up" operating area for configuring the softkeys in the Program Manager.

Information on the procedure and configuration can be found in the following documentation:

### References

Operating Manual, Universal/Turning/Milling/Grinding, Chapter "Managing programs" > "Setting up drives"

### File

The created configuration data is stored in the "logdrive.ini" file. This file is located in the /user/sinumerik/hmi/cfg directory.

## 3.3 Setting EXTCALL

Using the EXTCALL command, a program on an external program memory can be called from a part program.

EXTCALL does not need to be disabled.

### Activating the function

Processing EXTCALL calls is switched-on or switched-off using the following display machine data.

| MD9106 $MM_SERVE_EXTCALL_PROGRAMS | |
|---|---|
| = 0 | HMI ignores EXTCALL instructions and selection via PLC |
| = 1 | HMI processes EXTCALL instructions and selection via PLC (default value) |
| = 2 | HMI processes EXTCALL instructions and ignores selection via PLC |
| = 3 | HMI ignores EXTCALL instructions and processes selection via PLC |

**Note**

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.

## 3.4 Setting the EES

The EES (Execution from External Storage) function allows part programs for drives enabled for EES to be processed directly without reload buffer. The behavior is the same as that for processing from the NC part program memory without the restrictions that apply to "processing external".



**Software option**

In order to use this function, you require the "Execution from External Storage (EES)" software option.

For active EES function, a user memory (100 MB) on the system CF card is enabled. This allows the EES function to be used even without user CF card of the PPU.

**Note**

The EES function makes EXTCALL calls superfluous. If required, existing EXTCALL calls can be converted. EXTCALL calls without conversion are also executed for an active EES, although still as EXTCALL.

### Diagnostics

The EES mode is displayed using the following values to check the current EES setting:

| MD18045 $MN_EES_MODE_INFO | Mode, in which the EES function operates. |
|---|---|
| Bit 0 = 1 | Local EES active (LOCAL_DRIVE of the PPU) |
| Bit 1 = 1 | Global EES active (all available drives (USB, incl. local EES)) |
| Bit 2 = 1 | A global part program memory has been created in an external memory.<br>With the search path, the search is first performed for programs in the file system of the NC and then in the corresponding directories in an external memory. |

A detailed description of the EES function is provided in the following documentation:

### References

Function Manual, Basic Functions: Mode group, channel, program operation, reset response (K1)

Chapter "Execution from External Storage (EES)"

## 3.5 Creating a global part program memory

With the global part program memory (GDIR), a structure is provided analog to the passive file system of the NC. It is integrated in the search path of the NC for subprogram calls. If you create a GDIR, all participants in the system have access to the same logical drive. The participants can directly execute part programs from the common program memory. The GDIR so replaces or extends the part program memory of the NC.

**Note**

The global part program memory plays a role only for the EES function. The creation of a GDIR is, however, not essential for the EES operation.

### Possible procedure

1. Declare the global part program memory in the "Creating drives" window
2. Shift the content of the local program memory with SPF, MPF and WCS directory to an external drive, e.g. to a USB-FlashDrive.
3. Copy the directories that have been backed up to the global part program memory.

Additional information about managing the program memory is available in the following documentation:

### References

Work Preparation Programming Manual: File and Program Management

Chapter "Program memory"

## 3.6 Program selection

### 3.6.1 Displaying the softkey

The softkeys on the user interface for calling the program lists are activated via general configuration machine data.

## Configuring a softkey

| MD51041 $MNS_ENABLE_PROGLIST_USER | |
|---|---|
| 0 | No softkey |
| 1 | The "Prog. list" softkey is displayed |

| MD51043 $MNS_ENABLE_PROGLIST_MANUFACT | |
|---|---|
| 0 | No softkey |
| 1 | The "MANUFACTURER" softkey is displayed |

## Precondition

To create program lists, you require the following authorization:

- "Program manager" operating area: Access level 3 (password: User) or
- "Start-up" operating area: Access level 1 (password: Manufacturer).

## Program lists

You can create a program list by combining desired programs from the NC data, e.g. workpieces, part programs and subprograms. These programs can then be selected by the PLC for execution.

| Program lists | Directory |
|---|---|
| plc_proglist_user.ppl | /**user**/sinumerik/hmi/plc/programlist |
| plc_proglist_manufacturer.ppl | /**oem**/sinumerik/hmi/plc/programlist |

## Procedure




1. Select the "Start-up" operating area if you have the authorization for access level 1 (password: Manufacturer).

   - OR -




   Select the "Program manager" operating area if you have the authorization for access level 3 (password: User).



2. Press the menu forward key and the "Prog. list" softkey.
   The "Program List" window opens.







3. Press the "Manufacturer" softkey if you want to create a program list that can be used for all machines of a certain series.
4. Place the cursor in the desired line (program number).




5. Press the "Select program" softkey.
   The "Programs" window opens. The data tree of the NC memory with the directories "Workpieces", "Part programs" and "Subprograms" is displayed.




6. Place the cursor on the desired program and press the "OK" softkey.
   The selected program is inserted in the first line of the list together with its path.
   - OR -
   Enter the program name directly in the list.
   If you are making entries manually, check that the path is correct (e.g. //NC/WKS.DIR/MEINPROGRAMM.WPD/MEINPROGRAMM.MPF).
   If required, the extension (.MPF) is added.
   **Note:**
   A syntax check is not performed.
   With workpieces, make sure that there is a main program or a job list with the same name in the workpiece itself.




7. To remove a program from the list, place the cursor on the appropriate line and press the "Delete" softkey.
   - OR -




   To delete all programs from the program list, press the "Delete all" softkey.

# Configuring the network

# 4

## 4.1 Displaying the network overview

All the available network interfaces and their settings are listed in the "Overview of the Network Settings" window. For example, the IP addresses, subnet masks and MAC addresses for the company network are displayed here and can be processed.

### Procedure




1. Select the "Start-up" operating area.



2. Press the menu forward key.




3. Press the "Network" softkey.




The "Overview of the Network Settings" window opens.




4. Press the "Change" softkey to edit settings.

### See also

Settings of the factory network (Page 36)

Saving network settings (Page 37)

## 4.2 Settings of the factory network

The configuration for the network interface, active protocols and firewall exceptions of the company network are performed in the **Settings of the Factory Network** window.

| Setting | Description |
|---|---|
| Gateway | If this value is not empty, the host specified there is used as the default gateway, which means that all IP packets that cannot be directly assigned are sent here for routing. |
| DNS server | If DNS name servers are specified here (maximum three), then they must be used to resolve symbolic host names, i.e. at most of the points where an IP address is expected, a computer name can also be used instead.<br>The name server setting is also sent to your DHCP client (TCU, PG) via the DHCP server of the NCU, so that this can also work with symbolic names. |
| Time server | Here you can specify up to three NTP servers (UDP/123) that can be used by the NTPD on the NCU for time synchronization. |
| Host name | Here you can define a name for the local host. This manually assigned name takes priority over all other names.<br>The host name is determined in the following order:<br>• This entry (identical to the host name from the basesys.ini file)<br>• A name received from the DHCP server (reverse DNS lookup, i.e. which name belongs to the received IP address)<br>• A default name ("NONAME_...")<br>Because the host name is also used as DNS name, it must satisfy the corresponding requirements of the RFC:<br>• ASCII letters (a-z, A-Z), numbers (0-9) and '-'<br>• maximum 63 characters |
| Active protocols | Specifies which protocols are to be used in the company network. DCP (Discovery and Basic Configuration Protocol) and LLDP (Link Layer Discovery Protocol) are available for selection. |
| Firewall exceptions | You can specify ports here that are to be enabled in the external network interface. Port TCP/22 (SSH protocol) is always enabled. |
| • S7 communication (TCP/102) | Port for the S7 communication |

| Setting | Description |
|---|---|
| • VNC access (TCP/5900) | Port for the VNC access |
| • Additional ports | Any further ports can be enabled here. The entry starts with the protocol name (TCP or UDP), followed by a slash '/' and the port number of the service to be enabled. Several entries are separated by a space.<br>Example (enabling of the VNC servers): `TCP/5900 TCP/5904 TCP/5905` |

| NOTICE |
|---|
| **Security risk** |
| Note that opening the ports in the firewall can represent a security risk. Only enable the ports that you actually require. |
| Note that every accessible service can have security gaps. |

### Procedure




1. The network overview is open.




2. Press the "Company network" softkey.
   The "Settings of the Company Network" window opens.




3. Press the "Change" softkey to edit settings.

## 4.3 Saving network settings

The settings made in the NCU can be saved as a file.

A target folder in which the network configuration is to be saved, or an existing file that is to be overwritten with the new settings, can be selected in the **Save Network Settings: Select Archive** window.

### Procedure




1. The network overview is open and you have made network settings.




2. Press the "Save NCU settings" softkey.
   The "Save Network Settings: Select Archive" window opens.
3. Select the folder in which the network configuration is be stored. If a file is already available, it will be overwritten.
   - OR -
   Press the "New directory" softkey to create a new directory.

- OR -




Press the "Find" softkey, for example, to find the storage location of the network configuration file.




4. Press the "Find next" softkey if the file found during the search does not correspond to the required configuration file.




5. Press the "Cancel" softkey to reject the settings.

- OR -




Press the "OK" softkey to save the network configuration.

## 4.4 Station-related network diagnostics

With the aid of the station-related network diagnostics, you can find and examine faulty, incorrectly parameterized or unconnected network and bus interfaces.

The network/bus interfaces of the SINUMERIK 828D are:

- Company network (X130)
- Service (X127)
- PROFINET (PLC I/O interface PN1, PN2)
- PROFIBUS (integrated)

### 4.4.1 Displaying the PPU network interfaces

The network interfaces, e.g. company network X130 and service X127, and their availability are displayed in a tree topology in the "TCP/IP Diagnostics" window.

#### Detailed view

You can display detailed information about a selected interface, e.g. IP address, DNS name, etc.

#### Procedure




1. Select the "Diagnostics" operating area.



2. Press the menu forward key and the "Bus TCP/IP" softkey.






3. Press the "TCP/IP diagnostics" softkey.
   The "TCP/IP Diagnostics" window opens and displays the current availability of the network connections.
4. Position the cursor on the interface for which you wish to display the detailed information.




5. Press the "Details" softkey to display all available parameters of the configured network connections.




6. Press the "Details" softkey to hide the detailed display.




7. Press the "Return" softkey in order to exit the "TCP/IP Diagnostics" window and return to the "PROFINET/PROFIBUS Diagnostics" window.

## 4.4.2 TCP/IP diagnostics and configuration

The configuration of the network interfaces is displayed in the "TCP/IP Configuration" window.

### Service

You use the service interface X127 for start-up and diagnostics.

A PG/PC communicates with the control via this interface as peer-to-peer. The IP address 192.168.215.1 is permanently set for the service interface.

### Company network

The company network is used, for example, to access the network drives.

A PPU is connected to the company network via Ethernet interface X130.

### Availability of the network connections

### Network adapter connection



White Network cable inserted



Red Network cable not inserted

### Availability

The availability describes the percentage of faulty packages compared to all the sent and received packages.

Problems in the company network or the settling time during the start-up can cause fluctuations in the availability.



Green Greater than 95%



Yellow 50 - 95%



Red Less than 50%

**Details of the network connections**

- **Computer name**
  The computer name of the control.
- **DNS name**
  The name of the network adapter as stored in the DNS (Domain Name Service) server.
- **MAC address**
  The physical address of the network adapter.
- **Address type**
  Information about the configuration of the network adapter in the configuration file "basesys.ini":
  - DHCP: DHCP is active for this network adapter.
    The information behind the hyphen "-" specifies the DHCP mode of operation:
    *Off* - DHCP is deactivated for this adapter
    *Client* - a DHCP client that receives an IP address and further data from a server, is running on the interface
    *Server* - the SINUMERIK control provides a DHCP server at this interface that supplies clients on this network with IPs
    If no changes have been made for the network adapters in the configuration file, then "Default" is also output.
  - Manual
    The "Change" softkey allows manual settings, see "Changeable parameters" and "Address types" sections.

    **Note**

    In the "Change" mode, either "Manual" or "DHCP" can be selected (only for company network, X130).

- **IP address assigned**
  The current IP address of the network adapter.
- **Subnet mask assigned**
  The subnet mask of the network adapter.
- **DHCP server**
  The IP address of the DHCP server (for address type "DHCP").
- **State, DHCP server**
  State of the synchronous DHCP server:
  – Active: The current NCU is active in the group of NCUs in the "Synced Server" mode and transfers the IP addresses.
  – Standby: The NCU is not active as DHCP server. If the active DHCP server fails, this NCU can take over the task of theDHCP servers.
- **Mode, DHCP synchronous**
  Priority of the synchronous DHCP server:
  Low, high, master
- **DNS server 1, DNS server 2**
  The IP address of the DNS server.
- **Gateway**
  The IP address of the host that is to be assigned as gateway.

**Note**

All information that is not available is marked in the relevant table line with a hyphen "-".

### Modifiable parameters

The following parameters of the company network interface (X130) can be set:

- Address type
- IP address
- Subnet mask
- DNS server 1 and 2
- Gateway (applies for service interface and company network)

### Address types

- DHCP
  The values for the settable parameters IP address, subnet mask, DNS server 1, DNS server 2 and gateway are defined automatically by the DHCP server, in accordance with the entry in the "basesys.ini".
- Manual
  The values for the set parameters IP address, subnet mask, DNS server 1, DNS server 2 and gateway can be specified manually. The values are stored in "basesys.ini".

## Procedure

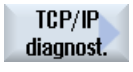



1. Select the "Diagnostics" operating area.



2. Press the menu forward key and the "Bus TCP/IP" softkey.




3. Press the "TCP/IP diagnostics" and TCP/IP config." softkeys.
   The "TCP/IP Configuration" window opens.







4. Press the "Modify" softkey to change specific parameters.
   The fields that can be edited have a white background.




5. Enter the desired changes and press the "OK" softkey to confirm the entries.




6. Press the "New display" softkey to refresh the display.

## 4.4.3 Ethernet network diagnostics

### 4.4.3.1 Diagnostics of the network and DHCP server

Press the "Network diagnostics" softkey to display data and availability of the network nodes. As result, the diagnostics provide, among other things, the following information:

- Component identification of the hardware
- TCP/IP address / MAC address
- DHCP server detection
- Status of the nodes

This gives you the option to test whether the current topology matches the reference topology.

## Procedure




1. Select the "Diagnostics" operating area.



2. Press the menu forward key and the "Bus TCP/IP" softkey.







3. Press the "TCP/IP diagnostics" softkey.

   The "TCP/IP Diagnostics" window opens and displays the current availability of the network connections.




4. Press the "Network diagnostics" softkey to browse your networks.

   The "Network Status" window opens and displays the addresses of the PN system network and the company network (X130).

   The corresponding fields in the window remain empty for non-accessible networks.




5. Press the "New display" softkey to refresh the display.

   - OR -




   Press the "DHCP server" softkey to fetch information and query the status of the DHCP server.

   The "DHCP Server" window opens.

   For each active DHCP server, the status, the role (priority), the MAC address, the TCU version and the lease version are displayed in tabular form in the upper part of the window. The required IP with the associated address area appears in the lower part of the window.

   Any differing data is displayed in red.

### 4.4.3.2 Accessible nodes

Pressing the "Accessible nodes" softkey creates and displays the list of nodes accessible via PROFINET. The list content can be compared as current topology with a reference topology.

**Note**

Although handheld terminals are not determined for the list of accessible nodes, they can be added as additional nodes to the topology.

## Procedure




1. The "Network Status" window opens.




2. Press the "Accessible nodes" softkey.
   The "Accessible Nodes" window opens.
   The name, IP address, HW ID/type, status and location are displayed in tabular form for each node.




3. Press the "Sort" softkey.













4. Press the "By name", "By IP", "By type" or "By location" softkeys to sort the display.

- OR -




3. If further information is provided for a selected node, press the "Details" softkey.
   The "Details" window opens and the additional information is displayed.







4. Press the "Zoom +" or "Zoom -" softkey until the desired display size is reached.

### Additional nodes







3. Press the "Additional nodes" softkey to search for nodes that are not listed under "Accessible nodes".

   The "Search for Additional Nodes" window opens. If necessary, change the address range and confirm with "OK".

   The found nodes are appended in the table; the HW-Id/type, Location, Contact and Description fields can then be edited and, by activating the checkbox, added to the first column, and, with a subsequent save, added to the reference topology.

### Comparing







3. Press the "Compare" softkey to compare the current list of nodes either with the previously saved manufacturer topology, a reference topology or a saved topology in the .xml format.

   The "Compare With" window opens. Position the cursor on the desired option box and confirm with "OK".

Deviations during the comparison are identified by being color-highlighted:

- Red: Missing or non-accessible components compared with the reference topology.
- Pink: Components that deviate in their attribute values compared with the reference topology.
- Blue: New components not contained in the reference topology.

### Saving







3. Press the "Save" softkey to save the current topology.

   The "Save As" window opens.

4. Position the cursor on the desired option box, specify a file name and possibly a comment, and confirm with "OK".

**Note**

Manufacturer topology and reference topology can be created with access level 3 (user) or higher and are always saved on the active DHCP server.

Default storage locations:

- Current topology (.xml or .csv):
  The following storage locations are offered depending on the configuration:
  - Network topology (storage location: HMI data / data backups / network topology)
  - User CF
  - Network drives
  - USB

  In CSV format, ";" is preset as a separator; it can be changed in sldgconfig.xml
- Reference topology:
  /user/common/tcu/$soll/$soll.xml
- Manufacturer topology:
  /user/common/tcu/$soll/$oem_soll.xml

# Customizing the "Machine" operating area 5

### Automatic changeover

The circumstances under which an automatic changeover is made into the "Machine" operating area are set in MD51040 $MNS_SWITCH_TO_MACHINE_MASK.

| MD51040 $MNS_SWITCH_TO_MACHINE_MASK | 0 (default setting) |
|---|---|
| Bit 0 = 1 | For program selection in the Program Manager, an automatic change is not made into the Machine operating area. |
| Bit 1 = 1 | For program selection in the Program Manager, an automatic change is not made into the Machine operating area.<br>When changing over the operating mode via the machine control panel, an automatic change is not made into the Machine operating area. |
| Bit 2 = 1 | For program selection in the Program operating area, an automatic change is not made into the Machine operating area. |
| Bit 3 = 1 | When selecting a program/executing a program in the Program operating area, the block search is not automatically started. |

## 5.1 Setting the font size of the actual value display

### Setting font size

The actual values for the machine coordinate system or workpiece coordinate system can be displayed in two different font sizes in the "Machine" operating area.

Using a channel machine data item, you can set how many axes should be displayed in a larger font.

| MD52010 $MCS_DISP_NUM_AXIS_BIG_FONT | Number of actual values with a large font |
|---|---|
| = 3 | Number of actual values with a larger font (default setting) |

## 5.2 Inserting a user-specific logo

### OEM-specific logo

The Siemens logo is visible in the header of the "Machine" operating area.

You can replace the Siemens logo with your own logo.

Procedure:

1. Name your own logo "`logo.png`".

   **Note**

   The file name must not contain any lower case letters.

2. Save your own logo depending on the bitmap size, in one of the directories: /**oem**/sinumerik/hmi/ico or /**user**/sinumerik/hmi/ico

| Directory | Bitmap size |
|---|---|
| /lco640 | 122 x 19 |
| /lco800 | 160 x 24 |
| /lco1024 | 199 x 30 |

## 5.3 Configuring the display of the G-code groups

The "G-functions" pane is visible in the "Machine" operating area.

Up to a maximum of 16 G-code groups are displayed in this pane. Configure the "slmagcodeconfig.ini" file accordingly to determine which G-code groups should be displayed, and where.

### Configuring G-code groups

| Section | Meaning |
|---|---|
| Selected G groups | This area is intended for operation of the controller in Siemens mode. |
| Selected G groups ISO mode | This area is intended for operation of the controller in ISO mode. |

Enter the G-code group and position as follows:

| SelectedGGroupN = M | |
|---|---|
| N | Position where G-code group M will be displayed.<br>If you want to leave a position empty, omit the number (M), e.g. "`SelectedGGroup1 = `" |
| M | The number of the G-code group to be displayed at position N |

### Structure of the file

```
[Selected G-Groups]
SelectedGGroup1 = 1
SelectedGGroup2 = 2
SelectedGGroup3 = 3
```

```
SelectedGGroup4 = 4
SelectedGGroup5 = 5
SelectedGGroup6 = 6
SelectedGGroup7 = 7
SelectedGGroup8 = 8
SelectedGGroup9 = 9
SelectedGGroup10 = 11
SelectedGGroup12 = 12
SelectedGGroup13 = 13
SelectedGGroup14 = 14
SelectedGGroup15 = 15
SelectedGGroup16 = 16
[Selected G-Groups ISO-Mode]
... (as with section for Siemens)
```

### Procedure

1. Copy the "slmagcodeconfig.ini" file from the folder /**siemens**/sinumerik/hmi/template/cfg.
2. Place the file in the following directory: /**oem**/sinumerik/hmi/cfg or /**user**/sinumerik/hmi/cfg. As soon as the file is located in the user-specific directory, the entries for this file take precedence over the Siemens file. If an entry is missing in a user-specific file, the corresponding entry from the Siemens file is used instead.
3. Open the file and enter the number of the G code in question at the required position "`SelectedGGroup1 =` **5**".
   I.e. the 5th G code group will be displayed at the 1st position.

## 5.4 Configuring the channel operational message

### Overview

The channel operational messages are displayed in the third line of the header in the "Machine" operating area. There are two types of channel operational messages:

1. Channel operational messages that require an operator action to continue a program, e.g. feed enable missing, M01 active.
   The channel operational messages are identified by exclamation marks.
2. Channel operational messages that do not require an operator action to continue a program. A stop condition is active for a length of time (>1 s) and is interrupted by the program processing, e.g. dwell time active, waiting for tool change.
   The channel operational message is identified by a clock icon and disappears after a certain time.

## File

The settings for this are specified in the "slmahdconfig.ini" file.

## Configuring the channel operational message

| Section | Meaning |
|---|---|
| CondStopIcons | Specification of an icon that is to be displayed with a certain channel operational message.<br>If no icon is specified for a certain channel operational message, then the icon specified for "DefaultIcon" is displayed. |
| CondDelayTime | Specification of how long (in milliseconds) a certain message must be present before it is displayed.<br>• If no time is specified for a certain channel operational message, then the time specified at "CondDelayTime" is used.<br>• If the channel operational message is to be displayed immediately, "0" must be specified as time. |

## Procedure

1. Copy the "slmahdconfig.ini" file from the /**siemens**/sinumerik/hmi/template/cfg directory.
2. Place the file in the following directory: /**oem**/sinumerik/hmi/cfg or /**user**/sinumerik/hmi/cfg.
3. Open the file and make the required settings.

## "slmahdconfig.ini" file

### Standard configuration

```
[CondStopIcons]
Default icon = condwait.png
1 =  condstop.png    ; No NC ready
2 =  condstop.png    ; No mode group ready
3 =  condstop.png    ; EMERGENCY STOP active
4 =  condstop.png    ; Alarm with stop active
5 =  condstop.png    ; M0/M1 active
6 =  condstop.png    ; Block in SBL mode terminated
7 =  condstop.png    ; NC stop active
8 =  condstop.png    ; Read-in enable missing
9 =  condstop.png    ; Feed enable missing
10 = condwait.png    ; Dwell time active
11 = condwait.png    ; Aux. function acknowledgment missing
12 = condstop.png    ; Axis enable missing
13 = condwait.png    ; Exact stop not reached
```

```
14 = condwait.png    ; Waiting for positioning axis
15 = condwait.png    ; Waiting for spindle
15 = condwait.png    ; Waiting for other channel
17 = condstop.png    ; Waiting for feedrate override
18 = condstop.png    ; Error in NC block
19 = condstop.png    ; Waiting for external NC blocks
20 = condwait.png    ; Waiting for synchronized action
21 = condwait.png    ; Block search active
22 = condstop.png    ; Spindle enable missing
23 = condstop.png    ; Axis feedrate override
24 = condwait.png    ; Waiting for tool change acknowledgment
25 = condwait.png    ; Gear step change
26 = condwait.png    ; Waiting for position control
27 = condwait.png    ; Waiting for thread cut
28 = condwait.png    ; Reserved
29 = condwait.png    ; Waiting for punching
30 = condwait.png    ; Waiting for safe operation
31 = condwait.png    ; No channel ready
32 = condstop.png    ; Oscillation active
33 = condwait.png    ; Axis replacement active
34 = condwait.png    ; Axis container rotation
35 = condwait.png    ; AXCT: Following axis active
36 = condwait.png    ; AXCT: Leading axis active
37 = condwait.png    ; AXCT: Follow-up active
38 = condwait.png    ; AXCT: Internal status change
39 = condwait.png    ; AXCT: Axis/spindle disable
40 = condwait.png    ; AXCT: Corr. motion active
41 = condwait.png    ; AXCT: Axis replacement active
42 = condwait.png    ; AXCT: Axis interpolator active
43 = condwait.png    ; Waiting for compile cycle
44 = condwait.png    ; Access to system variable
45 = condstop.png    ; Search target found
46 = condwait.png    ; Rapid retraction started
47 = condwait.png    ; AXCT: Wait for spindle stop
48 = condwait.png    ; Machine data match
49 = condwait.png    ; Axis replacement: Axis coupled
50 = condwait.png    ; Axis replacement: Liftfast active
51 = condwait.png    ; Axis replacement: New config active
52 = condwait.png    ; Axis replacement: AXCTSW active
53 = condwait.png    ; Axis replacement: Waitp active
54 = condwait.png    ; Axis in another channel
55 = condwait.png    ; Axis replacement: Axis is PLC axis
```

```
56 = condwait.png    ; Axis replacement: Axis is oscillating axis
57 = condwait.png    ; Axis replacement: Axis is Jog axis
58 = condwait.png    ; Axis replacement: Command axis
58 = condwait.png    ; Axis replacement: Axis is OEM axis
60 = condwait.png    ; Coupled following axis
61 = condwait.png    ; Coupled-motion following axis
62 = condwait.png    ; Coupled slave axis
63 = condstop.png    ; Stop at cycle end M0
64 = condstop.png    ; Stop at cycle end M1
65 = condwait.png    ; Wait: Axis is at fixed stop
66 = condwait.png    ; Master-slave switchover active
67 = condwait.png    ; Axis replacement: Axis is single axis
68 = condstop.png    ; Stop: Target reached after block search
69 = condwait.png    ; Synchronism: Synchronous spindle
70 = condwait.png    ; Deactivation position synch. spindle
71 = condwait.png    ; Waiting for transformer axis enable
72 = condstop.png    ; Waiting because of possible collision
73 = condstop.png    ; Jog: Position reached
74 = condstop.png    ; Jog: Direction blocked
75 = condwait.png    ; Brake request
76 = condwait.png    ; Wait: G4 still: U
77 = condstop.png    ; Axial feed disable from PLC
78 = condstop.png    ; Waiting for axial feed enable
79 = condwait.png    ; Axial feed disable from SYNACT
80 = condwait.png    ; Waiting for master spindle speed
81 = condwait.png    ; Waiting for parameter set change
82 = condwait.png    ; Waiting for end of motion before
                     transformation change
83 = condwait.png
84 = condwait.png
85 = condwait.png


[CondDelayTime]
Default delay time = 1000
1 = 0           ; No NC ready
2 = 0           ; No mode group ready
3 = 0           ; EMERGENCY STOP active
4 = 0           ; Alarm with stop active
5 = 0           ; M0/M1 active
6 = 0           ; Block in SBL mode terminated
7 = 0           ; NC stop active
8 = 0           ; Read-in enable missing
```

```
9 = 0           ; Feed enable missing
10 = 0          ; Dwell time active
12 = 0          ; Axis enable missing
16 = 0          ; Waiting for other channel
17 = 0          ; Waiting for feedrate override
18 = 0          ; Error in NC block
19 = 0          ; Waiting for external NC blocks
22 = 0          ; Spindle enable missing
23 = 0          ; Axis feedrate override
32 = 0          ; Oscillation active
45 = 0          ; Search target found
76 = 0          ; Wait: G64 still U
```

## 5.5 Deactivating program test

There are several program test options to test and run-in a part program that are executed using the "Program Test" function.

This function is active as standard. You can select the program test in the "Machine" operating area, in the "Program control" window and using the "PRT" check box.

### Deactivating the program test

You can deactivate the program test using the following configuration machine data. "PRT" is no longer displayed in the status line.

| MD51039 $MNS_PROGRAM_CONTROL_MODE_MASK | | Options for machine - program control |
|---|---|---|
| Bit 0 = 0 | Function program test not available | |

The test option is described in the following documentation:

### References

Function Manual, Basic Functions, Mode Group, Channel, Program Operation, Reset Behavior (K1)

## 5.6 Activating the "Teach In" function

### Activate TEACH IN

The TEACH IN function is activated in part via the general configuration machine data.

## Settings

| MD51034 $MNS_TEACH_MODE | |
|---|---|
| Bit 0 = 1 | The "Teach progr." softkey is displayed. The approached position is accepted into the program with the "Accept" softkey, default value. |
| Bit 1 = 1 | The acceptance of the teach-in block can be blocked by the PLC.<br>DB1700.DBX1000.0 = 0 Block is accepted<br>DB1700.DBX1000.0 = 1 Block is not accepted |

Access to the function still depends on access rights.

| MD51045 $MNS_ACCESS_TEACH_IN | Protection level TEACH IN |
|---|---|
| = 7 | The protection level for the execution of traversing movements to MDA buffer / part program for TEACH IN is specified, default value. |

| MD51050 $MNS_ACCESS_WRITE_PROGRAM | Protection level write part program |
|---|---|
| = 7 | The protection level for writing part programs is specified, default value. |

Set the channel-specific machine data:

| MD20700 $MC_REFP_NC_START_LOCK | NC-Start disable without reference point |
|---|---|
| = 0 | |

## Activating TEACH IN

1. Set the machine data and initiate a reset.
2. Now change into the "AUTO" and "MDA" operating mode.
3. Select the TEACH IN machine function.
4. Press the "Teach prog." softkey.

## TEACH IN not active

If the "Prog. teach in" softkey is not active, the reasons can be as follows:

- The program is not in the stop, reset or interrupt state.
- Not all of the axes that require referencing are referenced.

## Accept STAT and TU

If you wish to accept the joint position of the machine and the rotary axes in the traversing block, then activate the "STAT" and "TU" checkboxes.

In order that the checkboxes are displayed, set the following general configuration machine data:

| MD51032 $MNS_STAT_DISPLAY_BASE | | Numerical basis for display of the joint position STAT |
|---|---|---|
| = 1 | Display of the joint position STAT | |

| MD51033 $MNS_TU_DISPLAY_BASE | | Numerical basis, display of the rotary axes TU position |
|---|---|---|
| = 1 | Display of the rotary axes position TU | |

# 5.7 Block search

## 5.7.1 Activating block search mode

### Block search mode settings

The following machine data can be used to set the possible variants for the block search with G-code programming:

| MD51028 $MNS_BLOCK_SEARCH_MODE_MASK | Bit mask for the available search modes |
|---|---|
| Bit 0 = 1 | Block search with calculation, without approach |
| Bit 1 = 1 | Block search with calculation, with approach |
| Bit 4 = 1 | Block search without calculation |
| Bit 5 = 1 | Block search with test run |

The following machine data can be used to set the possible variants for the block search in a JobShop program (ShopMill, ShopTurn):

| MD51024 $MNS_BLOCK_SEARCH_MODE_MASK_JS | Bit mask for the available search modes (ShopMill, ShopTurn, single-channel) |
|---|---|
| Bit 0 = 1 | Block search with calculation, without approach (default setting) |
| Bit 1 = 1 | Block search with calculation, with approach<br>Notice:<br>In a ShopMill/ShopTurn program, the search is performed only on G code-blocks. |
| Bit 3 = 1 | Skip EXTCALL programs |
| Bit 5 = 1 | Block search with test run |

**Note**

The "Block search mode" softkey is only displayed, if, in the machine data MD51024 or MD51028, more than one version for the block search has been defined; for example "Block search with calculation without approach" and "Block search with calculation with approach".

When so doing, take into account that the search versions set active in the "Block search mode" window are also referred to the program.

### 5.7.2 Accelerated block search for execution from external

The accelerated block search function allows you to run the part program forward until you reach the block you require.

#### Settings

If you activate the "Skip EXCALL" function, then you must set the following general configuration machine data:

| MD51028 $MNS_BLOCK_SEARCH_MODE_MASK | Bit mask for the available search modes |
|---|---|
| Bit 3 = 1 (33H to 3BH) | |

For an external main program, a block search without calculation is always a fast block search. Only the blocks from the search position are loaded.

## 5.8 Manual machine

#### General information

"Manual machine" allows you to work with large cycles in manual mode without having to write programs especially for this purpose.



**Software option**

You require the following software option in order to use this function:

"ShopMill/ShopTurn"

#### Activation

| MD52212 $MCS_FUNCTION_MASK_TECH | | Cross-technology function mask |
|---|---|---|
| Bit 8 | Manual machine (ShopMill/ShopTurn) | |

#### Enables

| MD52214 $MCS_FUNCTION_MASK_MILL | | Milling function screen |
|---|---|---|
| Bit 5 | Enable spindle control, tool spindle | |
| = 0 | via the NC/PLC interface | |
| = 1 | via the user interface | |
| Bit 6 | Enable spindle control, rotating spindle | |
| = 0 | via the NC/PLC interface | |
| = 1 | via the user interface | |

| MD52218 $MCS_FUNCTION_MASK_TURN | | Turning function screen |
|---|---|---|
| Bit 3 | Enable spindle control, main spindle | |
| = 0 | via the NC/PLC interface | |
| = 1 | via the user interface | |
| Bit 4 | Enable spindle control, counterspindle | |
| = 0 | via the NC/PLC interface | |
| = 1 | via the user interface | |
| Bit 5 | Enable spindle control, tool spindle | |
| = 0 | via the NC/PLC interface | |
| = 1 | via the user interface | |

| | |
|---|---|
| SD43200 $SA_SPIND_S | Speed for spindle start using NC/PLC interface signal; value outputs using the operating software |
| SD43202 $SA_SPIND_CONSTCUT_S | Cutting speed for spindle start using NC/PLC interface signal; value outputs using the operating software |
| SD43206 $SA_SPIND_SPEED_TYPE | Spindle speed pipe for spindle start using NC/PLC interface signal; value outputs using the operating software |

## Additional parameter assignments

We recommend the following additional parameter assignments:

| Handwheel | |
|---|---|
| MD11346 $MN_HANDWH_TRUE_DISTANCE = 3 | Handwheel path or velocity specification |
| MD11602 $MN_ASUP_START_MASK bit 0, 3 =1 | Ignore stop conditions for ASUB |
| MD20150 $MC_G_CODE_RESET_VALUE [7] = 2 | Delete position of the G groups; run-up G54 active |
| MD20624 $MC_HANDWH_CHAN_STOP_COND = 0xFFFF | Definition of the behavior of traveling with handwheel, channel-specific |
| MD32084 HANDWH_STOP_COND = 0x7FF | Behavior, handwheel travel |
| MD52212 FUNCTION_MASK_TECH Bit 9 = 1 | Function screen form across technologies; work offset via softkey |

**Note**

If, in machine data MD52212 FUNCTION_MASK_TECH bit 9 is set, the "WO selection" softkey is displayed in the "Work Offset – G54 … G599" window.

| Working area limits | |
|---|---|
| MD28600 $MC_MM_NUM_WORKAREA_CS_GROUPS > 0 | Number of coordinate system-specific working area limits |
| MD20150 $MC_GCODE_RESET_VALUES[59] = 2 | Delete position of the G groups; activate working area limits of the group |

**Note**

If $MC_GCODE_RESET_VALUES[59] = 2 is set in the MD20150 machine data, the "Limit stops" softkey is displayed in the "Manual Machine" window. This requires that $MC_MM_NUM_WORKAREA_CS_GROUPS also has the value 1 for MD28600.

| Feedrate | |
|---|---|
| SD42600 $SC_JOG_FEED_PER_REV_SOURCE = -3 | Revolutional feedrate control in JOG mode |
| SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE [X axis] = -3 | Revolutional feedrate for positioning axis X |
| SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE [Z axis] = -3 | Revolutional feedrate for positioning axis Z |

**Note**

Ensure that the axis-specific setting data SD43300 SA_ASSIGN_FEED_PER_REV_SOURCE is set the same.

### References

- Operating:
  Operating Manual, Turning/Milling, Chapter "Working with a manual machine"
- Programming:
  Programming Manual, Fundamentals; Chapter "Working area limit"

## 5.9 Multi-channel support

In the "Machine" operating area, a dual channel view is possible for any two channels (2 from n) of a mode group.

Using the multi-channel editor, for a multi-channel machine, you can open, change or generate programs simultaneously that are independent of one another from a time perspective. In this case, the programs involved are displayed next to one another in the editor.

Further, you have the option of simulating the programs before they are actually processed at your control.

In addition, you can align the programs according to synchronization points.

**Note**

The multi-channel support is supported only for the turning and grinding technologies with the following CNC software variants:

- PPU 280/281 System Software Turning advanced
- PPU 280/281 System Software G-Tech advanced

## Requirement

- For multi-channel machine, the following data must be the same for all channels:
  - Measurement unit
  - Zero offset (e.g. G54)
  - Z value of the zero offset (optional)
  - Blank
  - Speed limitation
- Further, the following JobShop-specific data must be the same for all channels:
  - Retraction plane (one each for the main and counterspindle)
  - Tailstock (one each for the main and counterspindle)
  - Tool change point (one each for the main and counterspindle)
  - Safety clearance
  - Climbing and conventional

## Creating a multi-channel program

For multi-channel machining, you require a job list with the following data:

- Program names
- Program type
- G code
- ShopTurn program
- Channel assignment



**Software option**

You require the "ShopMill/ShopTurn" option to generate and edit ShopTurn/ ShopMill programs.

After you have created a job list, in the "multi-channel data" parameter screen, you can combine the data from the G code and JobShop, which is then saved as cycle call in the job list. The "multi-channel data" parameter screen is displayed as soon as at least one ShopTurn program is contained in a job list.

## Only programming G code programs

If the job list only contains G code programs, then you can deactivate the "multi-channel data" function. To do this, set the following machine data:

| MD51228 $MNS_FUNCTION_MASK_TECH | Cross-technology function screen |
|---|---|
| Bit 0 | G code programming without multi-channel data |
| = 1 | Multi-channel data is not listed for job lists that only contain G code programs. The "multi-channel data" softkey is not displayed. |

# Simulation and simultaneous recording 6

## 6.1 Simulation overview



**Software option**

You require the "3D simulation 1 (finished part)" option for the 3D representation of the simulation.

Simulation is supported with its own program interpreter (SNCK) and a separate simulation-data environment in SINUMERIK Operate. The SNCK considers the complete syntax of the SINUMERIK control family, including the possibility of incorporating special user options on the machine by comparing data with the NC environment. The simulation data can be matched statically as required with the NC environment (initialization data, macros, user data, tool data, machining cycles) or also dynamically when tool data or machining cycles are changed.

Machining simulations, with emphasis on the drilling, milling, turning and grinding technologies, can be performed in the workpiece coordinate system for certain machine kinematics on the user interface of the controller:

- The simulation of the finished part is performed with the real NC data.
- The NC data is automatically compared at each change of the part program.
- The simulation allows a dynamic representation of the machining, even for active 5-axis transformation (TRAORI) and swiveled planes.
- Turning on milling machines.
- Simulation of a machining channel for conventional milling machines.
- Simulation of up to four machining channels for turning machines with B axis.
- Intelligent determination of the block times and the program execution time.
- Very fast graphical representation through the continuous refinement of the workpiece.
- Optimum resolution for each selected picture area.
- Any sections are possible.
- In parallel to the machining of one workpiece, the machining of another workpiece can be simulated (as of NCU 720.x).
- 3D representation of simulation (option).

### General conditions for the simulation

- Very fast simulation result, this means, however, low dynamic resolution.
- For complex representations, fine recording is too slow or does not run for very complex parts.
- "Quickview" for mold-making part programs.
- The simulation cannot be used for compile cycles with special kinematics.

- Compile cycles are supported only to a limited extent.
- Blank CAD data cannot be imported (for example, for castings).
- Couplings cannot be simulated (coupled motion, axial master-value coupling, electronic gearbox):
  - Coupled motion: TRAILON(<following axis>, <leading axis>, <coupling factor>)
  - Axial master-value coupling: LEADON(Faxis, LAxis, curve table n) actual value and setpoint coupling
  - Electronic gearbox: EGDEF(FA, LA1, coupl.1, LA2, coupl.2, ... LA5, coupl.5), EGON, EGONSYN, EGONSYNE
- Reference point approach G74 is not operative during simulation
- The following is not supported:
  - Axis container
  - Dependencies on part programs and PLC
  - Multiple clamping
  - CNC function Synchronized actions for safe programmable logic (SPL) in the NC
- Different technologies in different channels with different axis coordinate systems are not fully supported.
- A 3D simulation is not possible for the grinding technology.
- The software limit switches are not evaluated:
  In a manufacturer cycle for the tool replacement, the "Software limit switch + the Z axis" machine data is used as replacement position:
  G0 Z=$MA_POS_LIMIT_PLUS[Z]-0.1
  The position, however, is not used in the simulation, but rather the initial setting of the machine data = 1e8 = 100,000,000. This specified position is traversed for the simulation. This makes the simulation very slow and causes the machining time to explode, e.g. 555 hours rather than 2 minutes.
- Program start in the simulation:
  If MD22622 $MCS_DISABLE_PLC_START is set in a channel, then the channel is not automatically started in the simulation. It can then only be started from another channel using the part program command "START". This function is not available for the SINUMERIK 828D.
- Alarm display in the simulation:
  Alarms are output in the simulation exactly the same way as alarms in the normal program execution, but no more than five alarms are displayed (above the graphic, to distinguish them from the machine alarms).
  If a different error signaling behavior (e.g. syntax error) occurs between simultaneous recording and simulation, then, for example, the reason can be due to the hardware access operations in the programming. The simulation does not access the hardware, however, internally it reads the input with 0 and is then exited. On the other hand, for simultaneous recording, an alarm is output.

# 6.2 Setting the technology for simulation

## Turning technology

Examples of machine kinematics:

- Conventional turning with two geometry axes
- Three spindles: Main spindle, counterspindle, tool spindle
- Counterspindle slides, tailstock as NC axis
- B axis: Aligning turning tools in the tool spindle
- Milling with geometry axes: TRANSMIT, TRACYL, TRAANG

Machine data for the turning technology:

| | |
|---|---|
| MD52200 $MCS_TECHNOLOGY = 1 | Turning technology |
| MD52000 $MCS_DISP_COORDINATE_SYSTEM = 34 | Position of the coordinate system (example) |

## Milling technology

Examples of machine kinematics:

- Milling with five axes: Swivel/TRAORI
- Swivel head change

Machine data for the milling technology:

| | |
|---|---|
| MD52200 $MCS_TECHNOLOGY = 2 | Milling technology |
| MD52000 $MCS_DISP_COORDINATE_SYSTEM = 0 | Position of the coordinate system (example) |

## Grinding technology

Machine data for the grinding technology:

| | |
|---|---|
| MD52200 $MCS_TECHNOLOGY = 3 | Cylindrical grinding technology |
| MD52200 $MCS_TECHNOLOGY = 4 | Surface grinding technology |
| MD52000 $MCS_DISP_COORDINATE_SYSTEM = 0 | Cylindrical grinding and surface grinding with moving column (example) |
| MD52000 $MCS_DISP_COORDINATE_SYSTEM = 6 | Surface grinding with table machine (example) |

## Meaning of the axes

| MD52206 $MCS_AXIS_USAGE[n] | Meaning of the axes in the channel<br>[n] channel axis number |
|---|---|
| = 0 | No special meaning |
| = 1 | Tool spindle (driven tool) |

| | |
|---|---|
| = 2 | Auxiliary spindle (driven tool) |
| = 3 | Main spindle (turning) |
| = 4 | C axis of the main spindle (turning) |
| = 5 | Counterspindle (turning) |
| = 6 | C axis of the counterspindle (turning) |
| = 7 | Linear axis of the counterspindle (turning) |
| = 8 | Tailstock (turning) |
| = 9 | Back rest (turning) |

Enter the direction of rotation for the rotary axes that are not configured in a toolholder or a 5-axis transformation via the following channel-specific machine data.

| MD52207 $MCS_AXIS_USAGE_ATTRIB[n] | Attributes of the axes<br>[n] channel axis number |
|---|---|
| Bit 0 | Rotation around 1st geometry axis (for rotary axes) |
| Bit 1 | Rotation around 2nd geometry axis (for rotary axes) |
| Bit 2 | Rotation around 3rd geometry axis (for rotary axes) |
| Bit 3 | Reversal of direction of rotation (for rotary axes) |

The MD52290 $MCS_SIM_DISPLAY_CONFIG machine data acts only on the OP019. It has no significance for SINUMERIK 828D.

| MD52290 $MCS_SIM_DISPLAY_CONFIG | Position of the status display of the channel in the simulation. |
|---|---|
| Bit 0 | Upper left-hand corner |
| Bit 1 | Upper right-hand corner |
| Bit 2 | Lower left-hand corner |
| Bit 3 | Lower right-hand corner |

In the channel-specific machine data, for at least one geometry axis per channel, enter a value other than 0.0, e.g. 0.001. If the value is 0.0, the system assumes that this parameter has still not been set.

| MD53230 $MCS_SIM_START_POSITION | Axis position when starting the simulation |
|---|---|
| Simulation is only possible if a value not equal to 0 has been set for at least one geometry axis. | |

## Deactivating simulation

The following machine data must be set to deactivate the simulation:

| **MD51226 $MNS_FUNCTION_MASK_SIM** | **Simulation function screen** |
|---|---|
| Bit 1 = 1 | Deactivate simulation |
| Bit 10 = 1 | Hide zero point symbol |

# 6.3 Simultaneous recording overview



**Software option**

You require the option "Simultaneous recording (real-time simulation)" for the "Simultaneous recording" function.



**Software option**

You require the "ShopMill/ShopTurn" option to display the tailstock



**Software option**

You require the "3D simulation 1 (finished part)" option for the 3D representation of the simultaneous recording function.

During machining, the tool paths can be simultaneously recorded on the display of the controller in a three-side view or 3D view. Workpiece graphics and views correspond to the graphic simulation.

**Note**

To prevent an incomplete representation, you must activate the simultaneous recording prior to the NC start.

### Example: Activating the tailstock representation in the spindle chuck data

Procedure:

1. Make the tailstock visible by making the selection in the program.
2. Specify the tailstock dimensions:
   Select in the operating area "Parameters" → "Setting data" → "Spindle chuck data", and enter the "ZR" and "XR" parameters for the tailstock length and tailstock diameter.
   The tailstock dimensions are written into the following machine data:

| MD52218 $MCS_FUNCTION_MASK_TURN | Turning function screen |
|---|---|
| Bit 2 = 1 | Tailstock enable:<br>The tailstock becomes visible if selected in the "tailstock" program. |

| MD53240 $MAS_ SPINDLE_PARAMETER[i] | Spindle chuck data in mm |
|---|---|
| [0]: Chuck dimension<br>[1]: Stop dimension<br>[2]: Jaw dimensions | |

| MD53241 $MAS_ SPINDLE_CHUCK_TYPE | Spindle jaw type |
|---|---|
| 0 = clamp from the outside<br>1 = clamp from the inside | |

| MD53242 $MAS_TAILSTOCK_PARAMETER[i] | Tailstock data in mm |
|---|---|
| [0]: Tailstock dimension<br>[1]: Tailstock length | |

### Activating the machining time display

Timers are provided as system variables in the Program runtime function. While the NC-specific timers are always activated (for time measurements since the last controller power-up), the channel-specific timers must be started via the following channel-specific machine data.

| MD27860 $MC_PROCESS_TIMERMODE | Activate the program runtime measurement. |
|---|---|
| Bit 0 = 1 | The measurement of the total runtime for all part programs is active ($AC_OPERATING_TIME). |
| Bit 1 = 1 | The measurement of the actual program runtime is active ($AC_CYCLE_TIME). |
| Bit 4 = 1 | Measurement also during active dry run feedrate. |
| Bit 5 = 1 | Measurement also during program test. |

## 6.4 Clamping a blank

The blank clamping function allows you to select and specify in more detail the clamping location for the blank.

You require this function if your machine satisfies one or more of the following requirements:

## Requirements

- The machine has several clamping options available for selection.
- The machine has a clamping option that does not point in the Z direction as standard (e.g. a boring mill).

Under the following circumstances, you can specify the position and orientation of the rotary axes in the machine data MD53220 and MD52207:

- The position and orientation of the rotary axis is offset against the MCS zero point, but not specified by a toolcarrier or a TRAORI.
- In a turning machine, the reference points of the main spindle or counterspindle are offset against the MCS zero point in X or Z.
- The counterspindle cannot be traversed with a saddle, but rather is at a fixed position.

In order that for simulation / simultaneous recording, the rotary axes are displayed at the correct location, their positions in the machine coordinate system must be specified using the following machine data:

| | |
|---|---|
| MD53220 $MAS_AXIS_MCS_POSITION[n] | Position of the axis in the MCS<br>[n] 0 = X, 1 = Y, 2 = Z |

For further information about the transformation of a rotary axis, see Chapter"Configuring cycles (Page 71)".

## Function

Set the following channel-specific machine data:

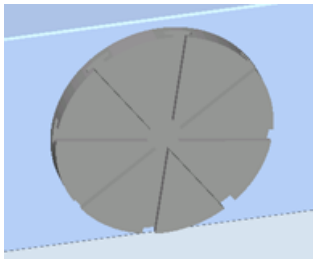
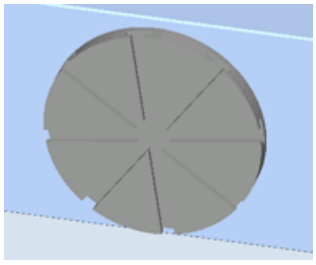
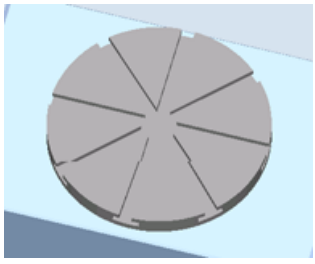
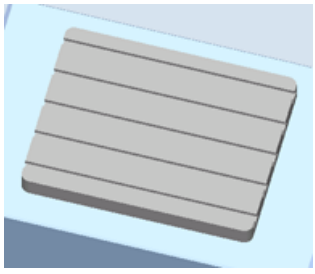
| MD52207 $MCS_USAGE_ATTRIB[n] | Attribute of the axes in the channel<br>[n] channel axis number |
|---|---|
| Bit 0 | Rotation around 1st geometry axis (for rotary axes) |
| Bit 1 | Rotation around 2nd geometry axis (for rotary axes) |
| Bit 2 | Rotation around 3rd geometry axis (for rotary axes) |
| Bit 8 | Offer rotary axis for blank clamping (on milling machines) |

| MD52214 $MCS_FUNCTION_MASK_MILL | Milling function screen |
|---|---|
| Bit 1 | Offer fixed table for blank clamping (on milling machines) |

Take the machine configuration for MD20080 from the example.

## Function options

In the program header, a blank can be optionally set to A, B, C and table clamping:

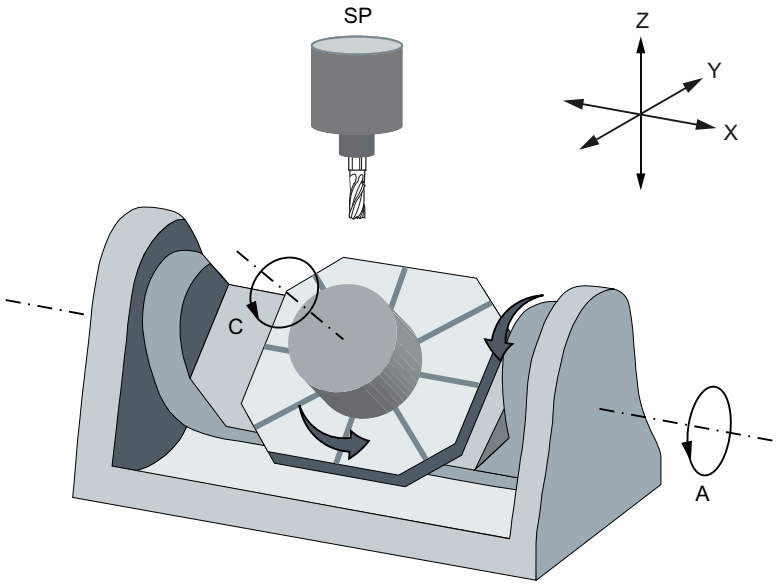
| Clamping | View | Machine data |
|---|---|---|
| A axis |  | MD52207 $MCS_USAGE_ATTRIB bit 0 and bit 8 = 1 |
| B axis |  | MD52207 $MCS_USAGE_ATTRIB bit 1 and bit 8 = 1 |
| C axis |  | MD52207 $MCS_USAGE_ATTRIB bit 2 and bit 8 = 1 |
| Table |  | MD52214 $MCS_FUNCTION_MASK_MILL bit 1 = 1 |

**Note**

To obtain the selection option in the program header, two rotary axes must be defined for the raw part clamping.

## Example for the machine configuration

Example for a machine configuration when the A axis should rotate at the X axis:




| MD20080 $MC_AXCONF_CHANAX_NAME_TAB[n] | Channel axis name in channel [n] |
|---|---|
| [0] | X |
| [1] | Y |
| [2] | Z |
| [3] | Spindle |
| [4] | A |
| [5] | C |

| MD52207 $MCS_USAGE_ATTRIB[4] | Attribute of rotary axis A in channel 4 |
|---|---|
| = 181H | |
| Bit 0 = 1 | Rotation around 1st geometry axis (for rotary axes) |
| Bit 8 = 1 | Offer rotary axis for blank clamping (on milling machines) |

## References

Operating Manual, Milling

- Chapter "Create ShopMill program" > "Program header"
- Chapter "Create ShopMill program" > "Change program settings"

# Configuring cycles 7

## 7.1 Activating turning/milling/drilling/grinding technologies

The cycles are automatically loaded when the control boots (Power On).

The configuration is realized using cycle machine data and cycle setting data. In so doing, a differentiation is made between the following data:

- General cycle machine data
- Channel-specific cycle machine data
- Axis-specific cycle machine data
- General cycle setting data
- Channel-specific cycle setting data
- Axis-specific cycle setting data

**Note**

**Specifying the technology**

For the SINUMERK 828D, the technology (turning, milling or grinding) is determined by the ordered software package. The technology cannot be parameterized with machine data.

### Adapting the user interface

The softkeys display for selecting the cycle support in the program editor is set by the following channel-specific configuration machine data:

| MD52200 $MCS_TECHNOLOGY | Technology |
|---|---|
| = 0 | Setting for turning |
| = 1 | Setting for turning |
| = 2 | Setting for milling |
| = 3 | Setting for cylindrical grinding technology |
| = 4 | Setting for surface grinding technology |

| MD52201 $MCS_TECHNOLOGY_EXTENSION | Technology extension for combined machines with several technologies |
|---|---|
| = 1 | Turning technology |
| = 2 | Milling technology |
| Example lathe with milling technology<br>MD52200 $MCS_TECHNOLOGY = 1<br>MD55201 $MCS_TECHNOLOGY_EXTENSION = 2 | |

| | |
|---|---|
| = 3 | Additional settings for the cylindrical grinding technology |
| = 4 | Additional settings for the surface grinding technology |

**References: grinding**

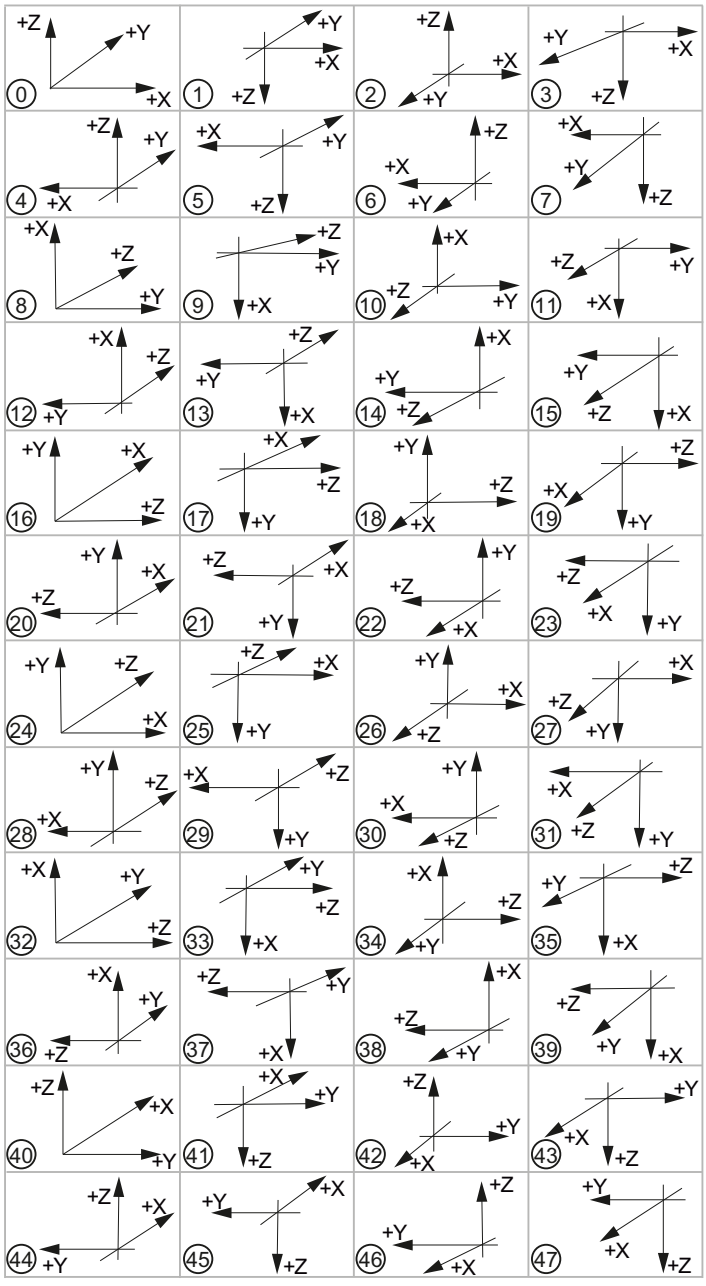
For further information about the grinding technology, please refer to:

- Chapter "grinding (Page 126)"
- Operating Manual, Grinding
- Programming Manual, Production Planning; Chapter "Programming cycles externally":
  - CYCLE495 - form-truing
  - CYCLE4071 - CYCLE4074: Longitudinal grinding cycles
  - CYCLE4075 - CYCLE4079: Surface grinding cycles
- Function Manual, Extended Functions; Chapter "Compensations (K3)" > "Interpolatory compensation" > "Cylinder error compensation"

## Setting the coordinate system

| MD52000 $MCS_DISP_COORDINATE_SYSTEM | Position of the coordinate system |
|---|---|
| = 0 | Default setting |
| With this machine data you adapt the position of the coordinate system of the user interface to the coordinate system of the machine. In the user interface, all help screens, sequence graphics, simulations, and input fields with circle direction indication change automatically according to the selected position. | |

The following settings can be selected for the coordinate system:




## Meaning of the axes

| MD52206 $MCS_AXIS_USAGE[n] | Meaning of the axes in the channel<br>[n] channel axis number |
|---|---|
| = 0 | No special meaning |
| = 1 | Tool spindle (driven tool) |
| = 2 | Auxiliary spindle (driven tool) |
| = 3 | Main spindle (turning) |
| = 4 | Separate C axis of the main spindle (turning) |

| | |
|---|---|
| = 5 | Counterspindle (turning) |
| = 6 | Separate C axis of the counterspindle (turning) |
| = 7 | Linear axis of the counterspindle (turning) |
| = 8 | Tailstock (turning) |
| = 9 | Back rest (turning) |
| = 10 | B axis (turning) |
| = 12 | B axis in counterspindle (turning) |
| = 13 | Transverse travel X of the counterspindle (turning) |

## Defining the direction of rotation

Enter the direction of rotation for the rotary axes that are not configured in a toolholder or a 5-axis transformation via the following channel-specific machine data.

| MD52207 $MCS_USAGE_ATTRIB[n] | Attribute of the axes in the channel [n] channel axis number |
|---|---|
| Bit 0 | Rotation around the 1st geometry axis (for rotary axes) |
| Bit 1 | Rotation around the 2nd geometry axis (for rotary axes) |
| Bit 2 | Rotation around the 3rd geometry axis (for rotary axes) |
| Bit 3 | Direction of rotation is counter-clockwise (for rotary axis / C axis). |
| Bit 4 | Displayed direction of rotation of the spindle / C axis for the M function M3 is counter-clockwise |
| Bit 5 | Inverts M3 / M4 (for spindles)<br>You must set this bit as for PLC bit DB38xx.DBX2001.6. |
| Bit 6 | Display rotary axis as offset target for measurement |
| Bit 7 | Offer rotary axis in the position pattern |
| Bit 8 | Offer rotary axis to accept blank (on milling machines) |
| Bit 9 | Spindle is not SPOS-capable |
| Bit 10 | Rotary axis rotates around the 1st geometry axis (only for position pattern) |
| Bit 11 | Rotary axis rotates around the 2nd geometry axis (only for position pattern) |
| Bit 12 | Rotary axis rotates around the 3rd geometry axis (only for position pattern) |

## Additional settings

| MD52005 $MCS_DISP_PLANE_MILL | PlaneSelectionMill |
|---|---|
| = 0 | Plane selection in the operator interface (cycle support for programming under "programGUIDE G-Code") |
| = 17 | G17 plane (default value) |
| = 18 | G18 plane |
| = 19 | G19 plane |

| MD52006 $MCS_DISP_PLANE_TURN | Plane selection turning |
|---|---|
| = 17 | G17 plane |

| | |
|---|---|
| = 18 | G18 plane (permanently set) |
| = 19 | G19 plane |

| MD52212 $MCS_FUNCTION_MASK_TECH | Cross-technology function screen |
|---|---|
| Bit 0 | Enable swivel with tool carrier |
| = 0 | Swivel plane, swivel tool not enabled |
| = 1 | Swivel plane, swivel tool enabled |
| Bit 1 | Optimized travel along software limit switches |
| = 0 | No optimized travel along software limit switches |
| = 1 | Optimized travel along software limit switches |
| Bit 5 | Call block search function SERUPRO. |
| = 0 | CYCLE207 is not called in the block search cycle PROG_EVENT.SPF. |
| = 1 | The cycle for SERUPRO (CYCLE207) is called in the block search cycle PROG_EVENT.SPF |

| MD52240 $MCS_NAME_TOOL_CHANGE_PROG | Tool change program for G code steps |
|---|---|
| = Program name | The associated program is called for tool change. |

# 7.2 Technology cycles for drilling

## Drilling technology

You can set drilling technology using the following channel-specific configuration machine data and channel-specific cycle setting data.

| MD52216 $MCS_FUNCTION_MASK_DRILL | Drilling function screen form |
|---|---|
| Bit 0 | Tapping cycle CYCLE84, technology input fields |
| = 0 | Hide input fields |
| = 1 | Display input fields |
| Bit 1 | Tapping cycle CYCLE840, technology input fields |
| = 0 | Hide input fields |
| = 1 | Display input fields |

| SD55216 $SCS_FUNCTION_MASK_DRILL_SET | Drilling function screen form |
|---|---|
| Bit 1 | Boring CYCLE86: Take into account the rotation of the tool plane when positioning the spindle<br>Note:<br>Spindle direction of rotation M3/ M4 and direction of rotation of the rotary axes must be set according to DIN. For SPOS=0, the tool cutting edge points in the + direction of the 1st axis of the plane (for G17, to X+). |
| = 0 | Do not take into account the rotation of the tool plane when positioning the spindle (SPOS). |
| = 1 | Take into account the rotation of the tool plane when positioning the spindle (SPOS). |

| | |
|---|---|
| Bit 2 | Boring CYCLE86: When positioning the spindle, take into account swiveled table kinematics (tool carrier) |
| = 0 | When positioning the spindle, do not take into account the components of the swiveled table kinematics (swiveling with CYCLE800, tool carrier).<br>Note:<br>Only the rotary axis of the machine kinematics that rotates the workpiece is taken into account. In the initial state of the kinematics, this rotary axis must be orientated parallel to the tool axis.<br>e.g. rotary axis C rotation around Z (for G17) rotary axis vector of the swivel data set V2xyz = 0 , 0 , -1. |
| = 1 | When positioning the spindle, take into account the components of the swiveled table kinematics (swiveling with CYCLE800, tool carrier). |
| Bit 3 | Tapping CYCLE84: Monitoring of the machine data MD31050 $MA_DRIVE_AX_RATIO_DENOM and MD31060 $MA_DRIVE_AX_RATIO_NUMERA of the spindle |
| Bit 4 | Tapping CYCLE84, set monitoring of the machine data MD31050 $MA_DRIVE_AX_RATIO_DENOM and MD31060 $MA_DRIVE_AX_RATIO_NUMERA of the spindle |
| = 0 | No monitoring |
| = 1 | Monitoring |
| Bit 6 | Boring CYCLE86: Adapt the spindle position to the tool orientation<br>Tool orientation can be realized using TOOLCARR or TRAORI before the cycle is called. |
| = 1 | Bits 1 and 2 are no longer effective. |
| Bit 7 | Boring CYCLE86: Adapt the direction of retraction travel in the plane to the active mirroring |
| = 0 | Retraction in the plane is realized as programmed. |
| = 1 | When mirroring is active, the direction when retracting in the plane is adapted in the cycle to the active mirroring. |

## Tapping (CYCLE84 and CYCLE840)

If the technology screens are hidden using the channel-specific configuration machine data MD52216 $MCS_FUNCTION_MASK_DRILL, then the settings in the following channel-specific cycle setting data are effective:

| SD55481 $SCS_DRILL_TAPPING_SET_GG12[0] | Setting tapping G group 12:<br>Block change behavior for exact stop |
|---|---|
| = 0 | Exact stop response as before the cycle call (default value). |
| = 1 | G601 |
| = 2 | G602 |
| = 3 | G603 |

| SD55482 $SCS_DRILL_TAPPING_SET_GG21[0] | Setting tapping G group 21:<br>Acceleration behavior |
|---|---|
| = 0 | Acceleration behavior as before the cycle call (default value). |
| = 1 | SOFT |
| = 2 | BRISK |
| = 3 | DRIVE |

| SD55483 $SCS_DRILL_TAPPING_SET_GG24[0] | | Setting tapping G group 24: Precontrol |
|---|---|---|
| = 0 | Precontrol, the same as before the cycle call (default value). | |
| = 1 | FFWON | |
| = 2 | FFWOF | |

### Tapping (CYCLE84)

| SD55484 $SCS_DRILL_TAPPING_SET_MC[0] | Spindle operation for MCALL |
|---|---|
| = 0 | For MCALL, reactivate spindle operation (default value). |
| = 1 | For MCALL, remain in position controlled spindle operation. |

## 7.3 Manufacturer cycles

### 7.3.1 Manufacturer cycles

The following cycles are also available in the cycle package for individual adaptation:

| | |
|---|---|
| CUST_TECHCYC.SPF | Manufacturer cycle for the adaptation of technology cycles. |
| CUST_800.SPF | Manufacturer cycle for the adaptation of the swivel plane and the swivel tool functions (CYCLE800), see Chapter "Manufacturer cycle CUST_800.SPF (Page 151)". |
| CUST_832.SPF | Manufacturer cycle for the adaptation of the high speed settings function (CYCLE832), see Chapter "Manufacturer cycle CUST_832.SPF (Page 160)". |
| CUST_MEACYC.SPF | Manufacturer cycle for the adaptation of the measuring functions, see Chapter "Manufacturer and user cycle CUST_MEACYC.SPF (Page 164)". |
| CUST_T | Cycle is used to track the T preparation after SERUPRO. |
| CUST_M6 | Cycle is used to track the tool change after SERUPRO. |
| CUST_MULTICHAN | Manufacturer cycle for lathes with several channels |
| CUST_CLAMP | Manufacturer's cycle for clamping elements |

### Copying manufacturer cycles

Copy the cycle from the NC data/Cycles/Standard cycles directory to the NC data/Cycles/User cycles directory before making any changes. Changes for the cycle take effect only in this directory.




1. Select the "Start-up" operating area.




2. Press the "System data" softkey.
3. Open the directory NC data/Cycles/Standard cycles and select the manufacturer cycles.




4. Press the "Copy" softkey and open the NC data/Cycles/Manufacturer cycles directory.




5. Press the "Paste" softkey.
   The cycles are available for you to individually adapt them.

## 7.3.2 PROG_EVENT.SPF standard cycle

### Rules for PROG_EVENT

PROG_EVENT.SPF is a standard cycle and cannot be changed (not accessible).

- Storage of a PROG_EVENT.SPF cycle that you have written under manufacturer or user cycles is **not** switched active.
- Manufacturer-specific "prog_events" must be reproduced by the following cycles, which, when available, are automatically called by the PROG_EVENT.
  - CYCPE_MA.SPF is called at the end of the internal PROG_EVENT.SPF.
    Note: CYCPE_MA.SPF should preferably be used.
  - CYCPE1MA.SPF is called at the beginning of the internal PROG_EVENT.SPF.
- Block search:
  Default setting: The correction of tool change, spindle direction of rotation / speed, swivel axes is performed by the internal PROG_EVENT, the necessary machine data is appropriately preset. The manufacturer does not have to use a CYCPE_MA.SPF for the actions described above.
  Special features of the block search in conjunction with PROG_EVENT:

| Machine data | | Bit |
|---|---|---|
| MD11450 | $MN_SEARCH_RUN_MODE | Bit 1=1 |
| MD52212 | $MCS_FUNCTION_MASK_TECH | Bit 3=1 |

**Note**

If $P_PROG_EVENT==5 is used in the CYCPE_MA.SPF, no REPOSA may be programmed as this is already present in the PROG_EVENT.SPF, otherwise an error occurs for SSL with calculation.

## 7.3.3 Manufacturer cycle for tool change CUST_T and CUST_M6

### Function

The two cycles are used to track the T preparation (CUST_T) and the tool change (CUST_M6) according to SERUPRO. In so doing, depending on the specific toolholder, the following is output:

- Last programmed change
- Last programmed preparation

### Sequence

The two cycles are called in the "SERUPRO-END-ASUB", which also provides the two transfer parameters.

| Parameter | Meaning |
|---|---|
| _THnr | Toolholder or master spindle number that was programmed for the change or the preparation. |
| _MTHnr | Number of the active toolholder or master spindle at the time of programming. |

The cycles themselves do not contain and "machine logic". Pure data operations are performed.

The active toolholder or the master spindle is saved.

Using the transfer parameter, the situation at the time of the T or M6 programming is determined. This situation is established, T and/or M06 output or a branch is made into the manufacturer cycles. After the output of T and M06 or after returning from the manufacturer cycles, the toolholder, saved at the beginning, is reactivated.

A change cycle can then only be called without any restriction, if the corresponding queries such as program test, block search,...exist.

### T replacement cycle

If a T replacement cycle is being used, then the following note must be observed:

**Note**

The CUST_T cycle uses the language command TCA (ToolChangeAbsolut) for the T call and therefore also has the possibility of inserting tools that have been disabled. The language command TCA is reconfigured to _TCA for 828D. TCA can be substituted by MD15710 $MN_TCA_CYCLE_NAME.

The following must be observed if working at a machine with T replacement:

- The T preparation cycle must be explicitly called at the specified position.
- If the call is realized using the CUST_T cycle, then the parameter evaluation ($C_TS_PROG, …) is skipped.
  Alternatively, the original TCA language command can be reprogrammed.

### References

Further information can be found in the "Function Manual Basic Functions", chapter: SERUPRO-End-ASUB

## 7.3.4 CUST_TECHCYC.SPF manufacturer cycle

### Function

For technological cycles, the CUST_TECHCYC cycle is called from ShopMill and ShopTurn cycles, if the machine builder must make certain adaptations to a specific machine (e.g. activate specific M commands). For instance, this may be necessary in order to retract or extend part catchers when cutting off parts on lathes or to position a spindle for clamping (only in JobShop).

The CUST_TECHCYC.SPF cycle can be used to program and execute the necessary sequences.

## Adapt manufacturer cycle CUST_TECHCYC.SPF

You can execute the following actions with the aid of the _MODE parameter:

| Marker | Action | |
|---|---|---|
| _M1 | Main spindle: | Changeover to C axis mode |
| _M2 | | Changeover to spindle mode |
| _M3 | | Clamp C axis |
| _M4 | | Release C axis clamping |
| _M5 | | Rinse chuck |
| _M6 | | Close chuck |
| _M7 | | Open chuck when spindle is stationary |
| _M8 | | Open chuck when spindle is rotating |
| _M9 | | Flush chuck |
| | | |
| _M11 | Tool spindle: | Changeover to C axis mode |
| _M12 | | Changeover to spindle mode |
| _M13 | | Clamp C axis |
| _M14 | | Release C axis clamping |
| | | |
| _M21 | Counterspindle: | Changeover to C axis mode |
| _M22 | | Changeover to spindle mode |
| _M23 | | Clamp C axis |
| _M24 | | Release C axis clamping |
| _M25 | | Rinse chuck |
| _M26 | | Close chuck |
| _M27 | | Open chuck when spindle is stationary |
| _M28 | | Open chuck when spindle is rotating |
| _M29 | | Rinse chuck off |
| | | |
| _M30 | 4. Position axis after block search (e.g. counterspindle, tailstock, back rest) | |
| | | |
| _M41 | Engage driven tool | |
| _M42 | Disengage driven tool | |
| | | |
| _M61 | Changeover to turning | |
| _M62 | Changeover to peripheral surface machining C | |
| _M63 | Changeover to face machining C | |
| _M64 | Changeover to peripheral surface machining Y | |
| _M65 | Changeover to face machining Y | |
| _M66 | Changeover to face machining B (swivel) | |
| _M67 | Deselection of the face machining B (swivel) | |
| _M68 | After block search to ST block before swivel | |
| | | |

| Marker | Action | |
|---|---|---|
| _M100 | Drawer: | Position before cut-off |
| _M101 | | Open during cut-off |
| _M102 | | Close after cut-off |
| _M103 | Cut-off completed | |
| | | |
| _M110 | Tool change | |
| _M111 | After end of tool change cycle (with tool change) | |
| _M112 | Before tool change on retraction plane | |
| _M113 | Before cutting edge reselection | |
| _M114 | After end of tool change cycle (without tool change) | |
| _M115 | After tool preselection (ShopTurn) | |
| _M116 | At tool change point (ShopTurn) | |
| _M117 | Before tool change T command | |
| | | |
| _M120 | Definition: Coupling of counterspindle / main spindle | |
| _M121 | After COUPOF switch off position control | |
| _M122 | Definition: Coupling, main/counterspindle | |
| _M123 | After COUPOF switch off position control | |
| _M125 | Changeover to main spindle | |
| _M126 | Changeover to counterspindle | |
| | | |
| _M131 | Start of program (ShopTurn program) | |
| _M132 | End of program header (ShopTurn program) | |
| _M135 | End of program loop (ShopTurn program) | |
| _M136 | End of program (ShopTurn program) | |
| | | |
| _M140 | Block search | Before tool change |
| _M141 | | After tool change |
| _M142 | | After tool change (new tool from ShopTurn) |
| | | |
| _M150 | 2-channel stock removal: | Leading channel before Balance Cutting |
| _M151 | | Leading channel after Balance Cutting |
| _M152 | | Following channel before Balance Cutting |
| _M153 | | Following channel after Balance Cutting |
| _M154 | | Start machining in the leading channel |
| _M155 | | End machining in the leading channel |
| _M156 | | Start machining in the following channel |
| _M157 | | Start machining in the following channel |
| | | |
| _M160 | Start multichannel programming | |
| _M170 | Start block with spindle | |
| _M171 | End block with spindle | |
| | | |

| Marker | Action |
|---|---|
| _M231 | Start of program (ShopMill program) |
| _M232 | End of program header (ShopMill program) |
| _M235 | End of program loop (ShopMill program) |
| _M236 | End of program (ShopMill program) |
| | |
| _M300 | Boring CYCLE86<br>Adapt the spindle to the tool position (SD55216 $SCS_FUNCTION_MASK_DRILL_SET bit 6 = 1) |

## 7.3.5 CUST_MULTICHAN user cycle

### Function

The CUST_MULTICHAN cycle is used for multi-channel programming at lathes. It is called at the start of the block when programming with blocks.

In the cycle, e.g. the master spindle is set to the spindle specified at the start of the block. The return value (= 1) can be used to control whether the complete block is skipped.

### CUST_MULTICHAN

| Parameter | Meaning |
|---|---|
| _S_NR | Spindle number to which the master spindle is set. |
| _RET | Return value |
| = 0 | The block is executed corresponding to the run-in mode. |
| = 1 | The entire block is skipped. |

## 7.3.6 Commissioning multi-channel machines with programSYNC

This section describes the special features that you must observe during the commissioning of a multi-channel machine with programSYNC (option) and ShopTurn.



**Software option**

You require the following software option in order to use the programSYNC function: "programSYNC".

## References

Basic Functions Function Manual

- Chapter "H2: Auxiliary function outputs to PLC > "Auxiliary functions"
- Chapter "K1: Mode Group, Channel, Program Operation, Reset Response" > "Block search"
- Chapter "K2: Axes, coordinate systems, frames" > "Frames"

## Setting the multi-channel view

In automatic operation, select under "Settings" → "Multi-channel function" which channels will be displayed adjacently.

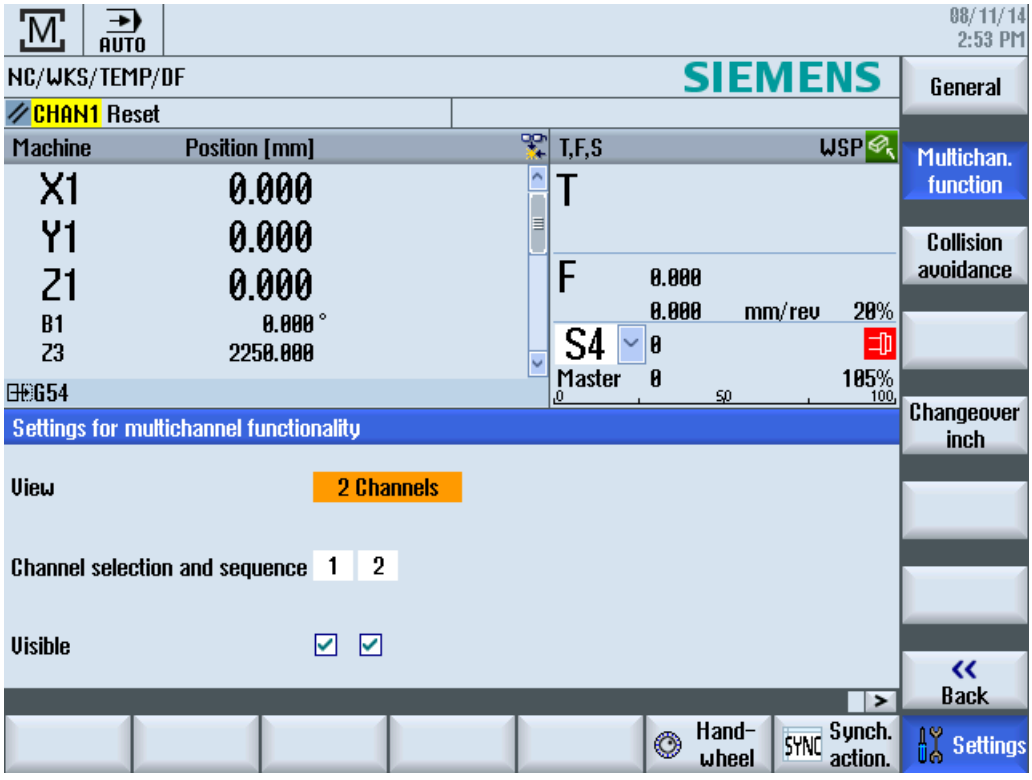



Figure 7-1 Dialog for viewing multiple channels

## Defining zero offsets

For multi-channel machines, it is desirable to maintain the same specific axes in certain zero offsets over all channels.

Both channels work on the same workpiece whose front edge normally has the value Z = 0 and which is defined using a set zero offset.

The position of the origin on the counterspindle is also defined using a set zero offset. This depends, however, also on the position of the Z3 motion axis of the counterspindle that exists just once.

Normally, assign the following frames for a zero offset:

| MD10616 $MN_MAPPED_FRAME_MASK | Frame mapping release |
|---|---|
| Bit 0 = 1 | System frame for actual value setting and scratching ($P_SETFR) |
| Bit 12 = 1 | Channel-specific base frames ($P_CHBFR) |
| Bit 13 = 1 | Set frames, zero offsets ($P_UIFR) |

With the following machine data, specify the connection of the axes that should have the same value in the zero offset over all channels:

| MD32075 $MA_MAPPED_FRAME[AXn] | Mapping of an axial frame |
|---|---|

**Example:**

Axis AX2 = Z1 (Z-axis in the first channel)

Axis AX6 = Z3 (motion axis of the counterspindle)

Axis AX11 = Z2 (Z-axis in the second channel)

MD32075 $MA_MAPPED_FRAME[AX6] = "AX6"

MD32075 $MA_MAPPED_FRAME[AX11] = "AX2"

The motion axis of the counterspindle is connected with itself (Z3 = Z3). The two Z-axes of the channels are connected (Z2 = Z1). You only need to create the connection of the axes in one direction.

## Program test run

To support the program test run, the following dialog is available under "Program control" → "Approach":

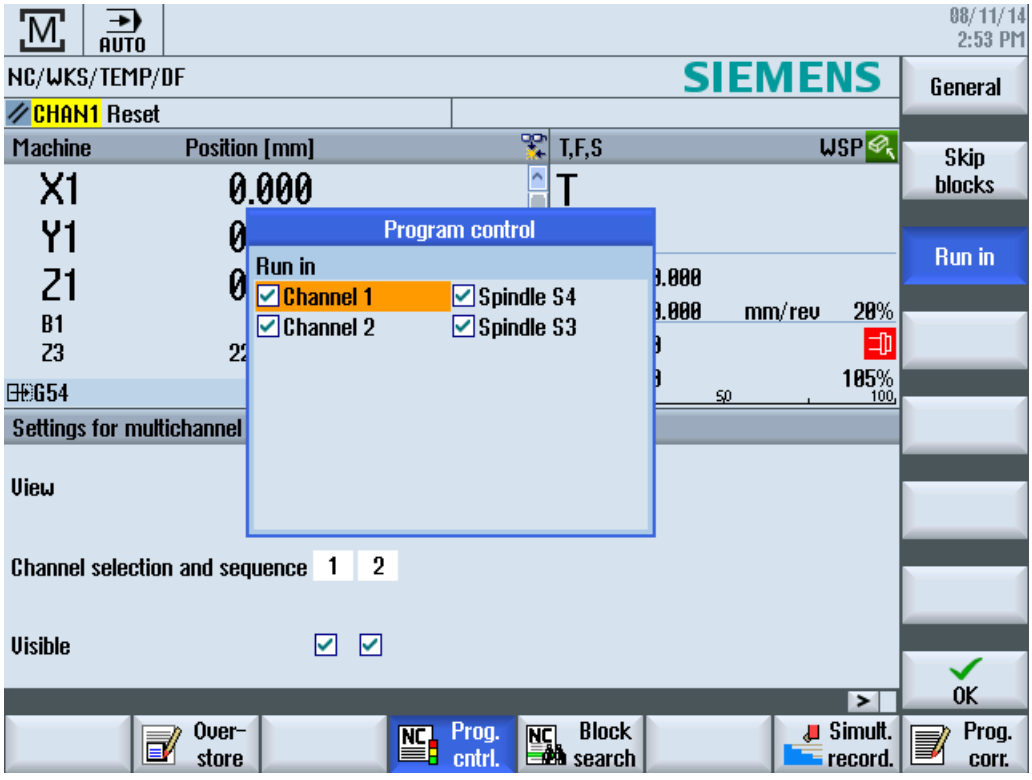



Figure 7-2 Program test run

In the dialog, specify which channel and which spindle should be used for the program test run.

The check marks for the program test run (dry run) are displayed as follows in the PLC:

(check mark = processing = no dry run = 0)

- Channel 1...n: DB330x.DBX1.7
- Spindle n: DB380x.DBX1.3

**Note**

An example for commissioning the positioning function is also supplied in the SINUMERIK 828D Toolbox.

All required M-commands must be grouped. Some groups are predefined for the standard M-commands (e.g. M3, M4, M5, ...), see following machine data item:

| | |
|---|---|
| MD22040 $MC_AUXFU_PREDEF_GROUP | Group assignment of predefined auxiliary functions. |

You must define user-defined M-commands in the following machine data item:

| | |
|---|---|
| MD22000 $MC_AUXFU_ASSIGN_GROUP | Auxiliary function group |

Bits 9, 10, 11 and 15 have a defined meaning in the following machine data:

| | |
|---|---|
| MD22035 $MC_AUXFU_ASSIGN_SPEC | Specification of the output behavior for the user-defined auxiliary functions. |
| MD22080 $MC_AUXFU_PREDEF_SPEC | Specification of the output behavior for the predefined auxiliary functions. |
| Bit 9 | Acquisition during block search type 5 (SERUPRO) |
| Bit 10 | No output during block search type 5 (SERUPRO) |
| Bit 11 | Cross-channel auxiliary function (SERUPRO) |
| Bit 15 | No output during the test run of the program (dry running) |

If a spindle is marked, the M-commands (direction of the spindle rotation, spindle chuck open/closed, ...) that relate to this spindle should be output. This must be initiated explicitly in the PLC (e.g. deactivate spindle via DB380x.DBX1.3).

If a channel is marked, M-commands (coolant, ...) that relate to this channel should be output. To do this, use bit 15. Alternatively, you can also select an initiation in the PLC.

Typical settings for cross-channel M-commands (e.g. M3, M4, M5) are:

Bit 9 = 1, bit 10 = 1, bit 11 = 1, bit 15 = 0

Typical settings for cross-specific M-commands (e.g. M7, M8, M9) are:

Bit 9 = 1, bit 10 = 1, bit 11 = 0, bit 15 = 1

## Calculating the program runtime

To calculate the program runtime, you can assign an average time to the M-commands in the following machine data:

| | |
|---|---|
| MD22037 $MC_AUXFU_ASSIGN_SIM_TIME | Acknowledgment time of the auxiliary function in msec. |

| | |
|---|---|
| MD22090 $MC_AUXFU_PREDEF_SIM_TIME | Acknowledgment time of the auxiliary function in msec. |

You can increase the time with the following commands:

- $AC_PRTIME_M_INC, operating time
- $AC_PRTIME_A_INC, motor stop time

You can consider individually known times.

## Block search with SERUPRO

To perform a fast block search, you must specify the following values:

| MD22600 $MC_SERUPRO_SPEED_MODE | Velocity for block search type 5 |
|---|---|
| = 0 | Program test with block search / dry run velocity: |

| MD22601 $MC_SERUPRO_SPEED_FACTOR | Velocity factor for block search type 5 |
|---|---|
| = 10000 | Factor which is multiplied by the dry run feedrate. |

These values are also required for the simulation.

To prevent M-commands from being issued during the search with SERUPRO, you must set bit 10 to "1" in the following machine data:

| MD22035 $MC_AUXFU_ASSIGN_SPEC | Specification of the output behavior for the user-defined auxiliary functions. |
|---|---|
| Bit 10 = 1 | No output during block search type 5 (SERUPRO) |

| MD22080 $MC_AUXFU_PREDEF_SPEC | Specification of the output behavior for the pre-defined auxiliary functions. |
|---|---|
| Bit 10 = 1 | No output during block search type 5 (SERUPRO) |

To start the part program, set the following machine data:

| MD10702 $MN_IGNORE_SINGLEBLOCK_MASK | Prevent single-block stop |
|---|---|
| Bit 16 = 1 | Single block has no effect with SERUPRO |

| MD10708 $MN_SERUPRO_MASK | Block change modes |
|---|---|
| Bit 0 = 1 | M0 has no effect with SERUPRO |
| Bit 4 = 1 | Override has no effect with SERUPRO |

To call the cycle for the SERUPRO process, set the following bit:

| MD52212 $MCS_FUNCTION_MASK_TECH | Cross-technology function screen |
|---|---|
| Bit 5 = 1 | Call block search function for SERUPRO. |

You can set the selection options for the search modes separately in the following machine data for programGUIDE and ShopTurn (single channel):

| MD51024 $MNS_BLOCK_SEARCH_MODE_MASK_JS | Bit mask for the available search modes (ShopMill, ShopTurn single channel) |
|---|---|
| Bit 0 = 1 | Block search with calculation, without approach |

| MD51028 $MNS_BLOCK_SEARCH_MODE_MASK | Bit mask for the available search modes |
|---|---|
| = 33H | |
| Bit 0 = 1 | Block search with calculation, without approach |
| Bit 1 = 1 | Block search with calculation, with approach |

| | |
|---|---|
| Bit 3 = 0 | Skip EXTCALL programs |
| Bit 4 = 1 | Block search without calculation (only programGUIDE) |
| Bit 5 = 1 | Block search with dry run |

## Block search: Position clamped axes

At the end of the PROGEVENT cycle, all axes are positioned automatically with REPOSA. Consequently, for clamped axes, the clamping must be opened automatically via the PLC or the axes must positioned by the machine manufacturer previously in the CYCPE_MA cycle.

**Example:**

```
IF ($P_PROG_EVENT==5)AND($P_SEARCHL==5); Block search with SERUPRO
    IF($AC_RETPOINT[B]<>$AA_IW[B]); Rotary axis is not in position
        M… ; Open clamping
        G0 B=$AC_RETPOINT[B]
        M… ; Reclamp axis
    ENDIF
ENDIF
```

## Block programming

It is preferable to use blocks during the multi-channel programming. To create a well structured program, you can combine machining blocks.

You can assign the blocks to a spindle. By default, the spindle becomes the master spindle at the start of the block. This behavior is configured in the CUST_MULTICHAN cycle and can be augmented by the machine manufacturer as required.

## Coordinate systems

Normally, the tools are used with 180° offset on the workpiece. This produces different coordinate systems for the channels:

- Channel 1

| MD52000 $MCS_DISP_COORDINATE_SYSTEM | Position of the coordinate system |
|---|---|
| = 34 | Z to the right, X upwards |

- Channel 2

| MD52000 $MCS_DISP_COORDINATE_SYSTEM | Position of the coordinate system |
|---|---|
| = 33 | Z to the right, X downwards |

You can also set this 180° offset in a base zero offset permanently for the C-axis of the second channel. Consequently, for example, holes at 0° are drilled in both channels at the same workpiece position.

### Tool list

The tool positions are displayed in accordance with the associated channel in the tool list. If only one TOA is available, non-loaded tools are always displayed in the coordinate system of the first channel.

### Main spindle and counterspindle

You must assign the spindles identically in both channels. If the main spindle in the first channel is the "left-hand" spindle, you must specify the main spindle in the second channel correspondingly.

You define the meaning of the axes in the channel in the following machine data, e.g. 3 = main spindle:

| MD52206 $MCS_AXIS_USAGE | Meaning of the axes in the channel |
|---|---|
| = 3 | Main spindle (turning) |

### Multi-channel data

A task list is created for the programming with programSYNC. This task list contains the multi-channel data that includes cross-channel information. The multi-channel data informs the simulation, e.g. about the raw part. The multi-channel data contains implicitly a WAIT flag to ensure that all programs in the involved channels are synchronized at the start.

- In ShopTurn programs, the multi-channel data is called automatically in the program header.
- In G-Code programs, the relevant call of the multi-channel data is added automatically via CYCLE208 after generating the task list.
  If this call should not be added automatically, you can suppress it with MD51228 $MNS_FUNCTION_MASK_TECH Bit 0 = 1 . The operator is then responsible for calling the CYCLE208 and for informing the simulation about the raw part.

### Program start

Two options are available to start programs in the channels:

- Press the "NC Start" key to start all channels automatically.
- Alternatively, the start can act only for the first channel that then starts all other involved channels.
  For these alternatives, you must adapt the "_M160" flag appropriately in the CUST_TECHCYC manufacturer cycle.

# 7.4 Milling

## 7.4.1 General

All settings for machine and setting data of the standard cycles are also applicable for milling under ShopMill and turning under ShopTurn.

## 7.4.2 Technology cycles for milling

### Milling function screen

| SD52214 $SCS_function_MASK_MILL | Milling function screen |
|---|---|
| Bit 0 | Enable cylinder surface transformation (ShopMill) |
| Bit 1 | List table to accept blank (on milling machines)<br>(This bit must only be set if the machine has a rotation axis and a fixed table for the blank clamping.) |
| Bit 2 | Special treatment of disc mill/saw with multi-edge<br>(For these tool types the first engagement is selected so that the top edge of the tool touches reference point Z0 exactly.) |
| Bit 3 | Enable machining inside/outside (ShopTurn) |
| Bit 4 | Enable clamp spindle (C-axis) (ShopTurn) |
| Bit 5 | Enable spindle control of the tool spindle via user interface |
| Bit 6 | Enable spindle control of the lathe spindle via user interface |

| SD55214 $SCS_FUNCTION_MASK_MILL_SET | Milling function screen |
|---|---|
| Bit 0 | Basic setting milling cycles in sync. |
| Bit 2 | Depth calculation of the milling cycles, with or without safety clearance. |
| = 0 | Depth calculation of the milling cycles is performed between the reference plane + safety clearance and the depth. |
| = 1 | Depth calculation is performed without including the safety clearance.<br>Bit 2 is effective in the following milling cycles: CYCLE61, CYCLE71, CYCLE76, CYCLE77, CYCLE79, CYCLE899, LONGHOLE, SLOT1, SLOT2, POCKET3, POCKET4. |

### Contour milling (CYCLE63)

| SD55460 $SCS_MILL_CONT_INITIAL_RAD_FIN | Finishing approach circle radius<br>The radius of the approach circle during the finishing of contour pockets is affected. |
|---|---|
| = 0 | The radius is selected so that at the starting point the safety clearance to the finishing allowance is maintained (default value). |
| > 0 | The radius is selected so that at the starting point the value of this channel-specific setting data to the finishing allowance is maintained. |

### Multiple edge (CYCLE79), circular position pattern (HOLES2), circumferential groove (SLOT2)

| SD55230 $SCS_CIRCLE_RAPID_FEED | Rapid feed in mm/min for positioning on a circular path between the circumferential grooves or the contour elements. |
|---|---|
| = 10000 | |

## 7.4.3 Setting-up ShopMill cycles for milling

### Load ShopMill cycles for milling

When the control boots, the ShopMill cycles are automatically loaded into the following directory: ../siemens/sinumerik/cycles/sm/prog

### Setting the coordinate system

| MD52000 $MCS_DISP_COORDINATE_SYSTEM | Position of the coordinate system |
|---|---|
| = 0 | |

**Examples:**

=0 for vertical milling machines

=16 for horizontal milling machines, boring mill

A detailed description of the MD52000 is provided in Section:

Activating turning/milling/drilling/grinding technologies (Page 71)

For plane selection when milling, appropriately set the following machine data:

| MD52005 $MCS_DISP_PLANE_MILL | Plane selection for milling |
|---|---|
| = 17 | |
| = 0 | In the cycle support |
| = 17 | G17 (default value) |
| = 18 | G18 |
| = 19 | G19 |

### Simulation and simultaneous recording

In order that simulation and simultaneous recording are displayed without any errors, set the machine data as described in the following section:

Setting the technology for simulation (Page 63)

## Defining the direction of rotation

In order that the direction of rotation is correctly displayed in the ShopMill user interface, and when programming ShopMill functions, the correct direction of rotation is executed, you must make some settings that are coordinated with one another. You must align these settings to the actual direction of rotation of the axis at the machine.

| MD52207 $MCS_USAGE_ATTRIB[n] | Attribute of the axes in the channel<br>[n] channel axis number |
|---|---|
| Bit 4 | Displayed direction of rotation for M3 is counter-clockwise (for spindles) |
| Bit 5 | Direction of rotation M3 corresponds to rotary axis minus (for spindles)<br>This bit must be set analog to PLC bit DB38xx.DBX2001.6. |

## Additional channel-specific machine data

| MD52216 $MCS_FUNCTION_MASK_DRILL | Drilling function screen form |
|---|---|
| Bit 2 | Tapping also without encoder<br>The setting depends on whether at the machine the tool spindle is equipped with an encoder. |
| = 0 | Tapping only possible with encoder |
| = 1 | Tapping also possible without encoder |
| Bit 3 | Constant cutting speed referred to the diameter of the centering |
| = 0 | Constant cutting speed referred to the diameter of the tool |
| = 1 | Constant cutting speed referred to the diameter of the centering |

| MD52229 $MCS_ENABLE_QUICK_M_CODES | Enable fast M commands |
|---|---|
| = 0 | |
| Bit 0 | Coolant OFF (default value) |
| Bit 1 | Coolant 1 ON |
| Bit 2 | Coolant 2 ON |
| Bit 3 | Coolants 1 and 2 ON |

| MD52230 $MCS_M_CODE_ALL_COOLANTS_OFF | M code for all coolants Off |
|---|---|
| = 9 | You define the M function to switch off the coolant that is output when the tool is changed. |

| MD52231 $MCS_M_CODE_COOLANT_1_ON | M code for coolant 1 On |
|---|---|
| = 8 | You define the M function for coolant 1 that is output when the tool is changed. |

| MD52232 $MCS_M_CODE_COOLANT_2_ON | M code for coolant 2 On |
|---|---|
| = 7 | You define the M function for coolant 2 that is output when the tool is changed. |

| MD52233 $MCS_M_CODE_COOLANT_1_AND_2_ON | M code for both coolants On |
|---|---|
| = -1 | You define the M function for coolant 1 and 2 that is output when the tool is changed. |

| MD52281 $MCS_TOOL_MCODE_FUNC_ON[ ] | M code for tool-specific function ON |
|---|---|
| = -1 | M function is not output.<br>If both M commands of a function "= -1", the corresponding field is not displayed on the interface. |
| [0] | M code for tool-specific function 1 ON |
| [1] | M code for tool-specific function 2 ON |
| [2] | M code for tool-specific function 3 ON |
| [3] | M code for tool-specific function 4 ON |

| MD52282 $MCS_TOOL_MCODE_FUNC_OFF[ ] | M code for tool-specific function OFF |
|---|---|
| = -1 | M function is output.<br>If both M commands of a function "= -1", the corresponding field is displayed on the interface. |
| [0] | M code for tool-specific function 1 OFF |
| [1] | M code for tool-specific function 2 OFF |
| [2] | M code for tool-specific function 3 OFF |
| [3] | M code for tool-specific function 4 OFF |

### Channel-specific cycle setting data

| SD55212 $SCS_FUNCTION_MASK_TECH_SET | Cross-technology function screen |
|---|---|
| = 6 | |
| Bit 0 | Tool preselection active.<br>The next tool is prepared directly after a tool change.<br>Notice: For a revolver, the setting data must be set to "0". |
| Bit 1 | Automatic calculation of the thread depth for metric threads. |
| Bit 2 | Take the thread diameter and thread depth from the table. |

## 7.4.4 Cylinder surface transformation (TRACYL)



**Software option**

You require the following software option in order to use this function:
"Transmit and peripheral surface transformation".

### Precondition

- There must be at least one rotary axis at the machine.
- The milling tool must be radially oriented to the cylinder to be machined.

### Function

The following groove machining operations can be performed with the Cylinder surface transformation functions:

- Longitudinal grooves on cylindrical bodies
- Transverse grooves on cylindrical objects
- Grooves with any path on cylindrical bodies

The path of the grooves is programmed with reference to the unwrapped, level surface of the cylinder. Programming can be realized using straight line/circle, drilling or milling cycles or contour milling (free contour programming).

There are two variants of cylinder surface transformation, i.e.

1. with groove side offset (ON)
2. without groove side offset (OFF)

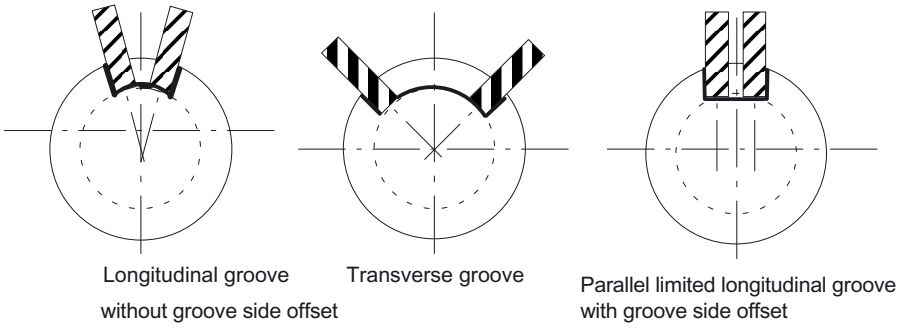


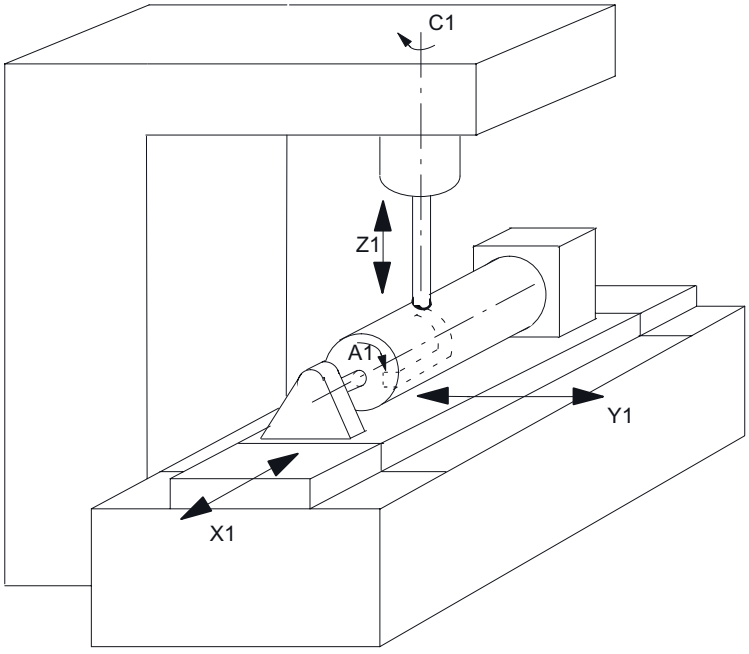

Figure 7-3 Grooves with and without groove side offset

## 7.4.5 Example: Milling machine with the XYZ-AC axis configuration

### Example: Milling machine with axis configuration X1Y1Z1-A1C1

Example of an axis configuration on the following machine:

## Axis configuration




X1 1. axis of the machining plane parallel to the rotary axis

Y1 2. axis of the machining plane

Z1 Infeed axis (tool axis) perpendicular (radial) to the rotary axis

A1 Rotary axis

C1 Working spindle

You must configure two data records with the following machine data for the machine illustrated above:

| MD20070 $MC_AXCONF_MACHAX_USED[4] | Machine axis number valid in channel |
|---|---|
| = 5 | Number of channel axes |

| MD20080 $MC_AXCONF_CHANAX_NAME_TAB[ ] | Name of channel axis in the channel |
|---|---|
| [0] = X1 | channel axis X1, corresponds to channel axis 1 |
| [1] = Y1 | channel axis Y1, corresponds to channel axis 2 |
| [2] = Z1 | channel axis Z1, corresponds to channel axis 3 |
| [3] = A1 | channel axis A1, corresponds to channel axis 4 |
| [4] = C1 | channel axis C1, corresponds to channel axis 5 |

| MD28082 $MC_MM_SYSTEMFRAME_MASK | System frames (SRAM) |
|---|---|
| Default, cannot be changed | |

## General settings for the transformation

| MD10602 $MN_FRAME_GEOAX_CHANGE_MODE | | Frames when switching over geometry screens |
|---|---|---|
| = 1 | The actual total frame (zero offsets) is recalculated when switching over geometry axes (selecting - deselecting TRACYL). | |

## Data set for the 1st transformation in the channel

| MD24100 $MC_TRAFO_TYPE_1 | | Definition of transformation 1 in the channel. |
|---|---|---|
| = 512 | Transformer type (512 = cylinder surface transformation without groove side offset) | |

| MD24110 $MC_TRAFO_AXES_IN_1 | | Axis assignment for the 1st transformation in the channel |
|---|---|---|
| [0] = 3 | Channel axis: Infeed axis (tool axis) perpendicular (radial) to the rotary axis | Z |
| [1] = 4 | Channel axis: Rotary axis | A |
| [2] = 1 | Channel axis: 1st axis of the machining plane parallel to the rotary axis | X |
| [3] = 2 | Channel axis: 2nd axis of the machining plane | Y |

| MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 | | Assignment of the geometry axes to channel axes for transformation 1. |
|---|---|---|
| [0] = 1 | Channel axis: 1st geometry axis | X |
| [1] = 4 | Channel axis: 2nd geometry axis | A |
| [2] = 3 | Channel axis: 3rd geometry axis | Z |

| MD24800 $MC_TRACYL_ROT_AX_OFFSET_1 | |
|---|---|
| = 0 | Offset of rotary axis for the 1st TRACYL transformation. |

| MD24805 $MC_TRACYL_ROT_AX_FRAME_1 | |
|---|---|
| = 1 | Axial offset of rotary axis for the 1st TRACYL transformation. |

| MD24810 $MC_TRACYL_ROT_SIGN_IS_PLUS_1 | |
|---|---|
| = 1 | Sign of the rotary axis for the 1st TRACYL transformation. |

| MD24820 $MC_TRACYL_BASE_TOOL_1[ ] | |
|---|---|
| [0] = 0 | Vector of basis tool for the 1st TRACYL transformation in XYZ |
| [1] = 0 | |
| [2] = 0 | |

## Data set for the 2nd transformation in the channel

| MD24200 $MC_TRAFO_TYPE_2 | |
|---|---|
| = 513 | Transformer type (513 = cylinder surface transformation with groove side offset) |

| MD24210 $MC_TRAFO_AXES_IN_2 | | |
|---|---|---|
| [0] = 3 | Channel axis: Infeed axis perpendicular (radial) to rotary axis | Z |
| [1] = 4 | Channel axis: Rotary axis | A |
| [2] = 1 | Channel axis: 1st axis of the machining plane parallel to the rotary axis | X |
| [3] = 2 | Channel axis: 2nd axis of the machining plane | Y |

| MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2 | | |
|---|---|---|
| [0] = 1 | Channel axis: 1st geometry axis | X |
| [1] = 4 | Channel axis: 2nd geometry axis | A |
| [2] = 3 | Channel axis: 3rd geometry axis | Z |

| MD24850 $MC_TRACYL_ROT_AX_OFFSET_2 | |
|---|---|
| = 0 | Offset of rotary axis for the 2nd TRACYL transformation |

| MD24855 $MC_TRACYL_ROT_SIGN_IS_FRAME_2 | |
|---|---|
| = 1 | Offset of rotary axis for the 2nd TRACYL transformation |

| MD24860 $MC_TRACYL_ROT_SIGN_IS_PLUS_2 | |
|---|---|
| = 1 | Sign of rotary axis for the 2nd TRACYL transformation |

| MD24870 $MC_TRACYL_BASE_TOOL[ ] | |
|---|---|
| [0] = 0 | Vector of basis tool for the 2nd TRACYL transformation in XYZ |
| [1] = 0 | |
| [2] = 0 | |

**Note**

**Data records for transformations**

For both data records, you can use any transformations from all available transformations (MD24100 $MC_TRAFO_TYPE_1, MD24200 $MC_TRAFO_TYPE_2, etc.). The two data records need not be directly next to each other.

However, the 1st data record must always be used for "Cylinder surface transformation **without** groove side offset" (= 512) and the 2nd data record for "Cylinder surface transformation **with** groove side offset" (=513).

## 7.4.6 ShopMill cycles for multiple clamping



**Software options**

You require the following software option in order to use this function: "ShopMill/ ShopTurn"

### Function

The Multiple clamping function provides optimized tool change over several workpiece clampings. On the one hand, this reduces downtimes and, on the other, eliminates tool change times because a tool performs as many machining operations as possible in all clampings before the next tool change is initiated.

You can either execute the same program several times for the clampings or you can select different programs. If you are using large-area fixture plates on your machine, you do not have to set up anything else.

In the case of rotating clamping devices, on the other hand, you must adapt a cycle to match the features of the clamping device so as to ensure that the next workpiece can be turned to the machining position after machining of the previous one (or for multiple clamping devices even while the current workpiece is being machined).

### Procedure for setting up

1. Change the CLAMP.SPF cycle which is stored in the cycle/sc/prog/templates_deu or cycle/sc/prog/templates_eng directory.
2. Copy the cycle into the user or manufacturer directory.

### Example

A rotating clamping device (reversible clamping device) is used with four clampings. The respective clampings can be machined by positioning rotary axis A:

| | |
|---|---|
| Clamping 1: | A = 0° |
| Clamping 2: | A = 90° |
| Clamping 3: | A = 180° |
| Clamping 4: | A = 270° |

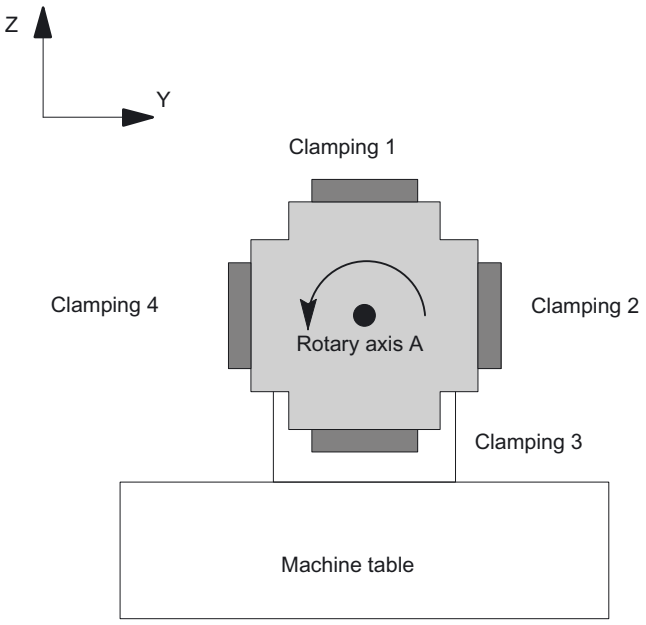



Figure 7-4 Rotating clamping device (reversible clamping device)

The cycle CLAMP.SPF must be adapted as follows:

```
...
DEF INT _NV ; Auxiliary variable
;-----------------------------------
;Adaptation
;
IF _ACT==1
G0 A=DC(0)
ENDIF
;
IF _ACT==2
G0 A=DC(90)
ENDIF
;
IF _ACT==3
G0 A=DC(180)
ENDIF
;
IF _ACT==4
G0 A=DC(270)
```

```
ENDIF
;
;-----------------------------------
_NV=_NPV+_ACT ;  Calculate current zero offset
N10 G[8]=_NV ;  There must not be a calculation here
RET
```

# 7.5 Turning

## 7.5.1 General

All settings for machine and setting data of the standard cycles are also applicable for milling under ShopMill and turning under ShopTurn.

## 7.5.2 Setting up cycles for turning

### Setting the coordinate system

| MD52000 $MCS_DISP_COORDINATE_SYSTEM | Position of the coordinate system |
|---|---|
| = 0 | Default setting |

Examples:

0: Vertical lathe (carousel-type lathe)
19: Horizontal lathe, machining in front of the turning center
34:  Horizontal lathe, machining behind the turning center (inclined bed lathe)

A detailed description of the MD52000 is provided in Section: "Activating turning/milling/drilling/grinding technologies (Page 71)".

### Turning function screen

Enable various functions under the Turning function screen in the following channel-specific configuration machine data.

| MD52218 $MCS_FUNCTION_MASK_TURN | Turning function screen |
|---|---|
| Bit 0 | Enable zoom under manual for tool measurement |
| Bit 1 | Enable part catcher for cut-off |
| Bit 2 | Enable tailstock |
| Bit 3 | Enable spindle control of main spindle via user interface |
| Bit 4 | Enable spindle control of the counterspindle control via user interface |

| | |
|---|---|
| Bit 5 | Enable spindle control of the tool spindle via user interface |
| Bit 6 | Enable balance cutting for two-channel stock removal |
| Bit 7 | Retraction when cutting along the contour with G1 |
| Bit 8 | Enter the spindle chuck data in the program |
| Bit 9 | Additional input of the tailstock data in the program |
| Bit 12 | Deactivate thread synchronization |
| Bit 13 | Cutting along the contour with CYCLE95 (828D programGUIDE without Advanced Technology |

## Meaning of the axes

You set the meaning of the axes in machine data MD52206 $MCS_AXIS_USAGE[n].

| MD52206 $MCS_AXIS_USAGE[n] | Meaning of the axes in the channel<br>[n] channel axis number |
|---|---|

A detailed description of MD52206 is provided in Chapter "Activating turning/milling/drilling/grinding technologies (Page 71)

## Traversing direction

In order that the user interface and cycles know in which direction the spindles and rotary axes rotate, the following settings must be made.

You set the direction of rotation of an NC rotary axis via MD32100 $MA_AX_MOTION_DIR.

| MD32100 $MA_AX_MOTION_DIR | Traversing direction |
|---|---|
| -1 | Direction reversal |
| 0, 1 | No direction reversal |

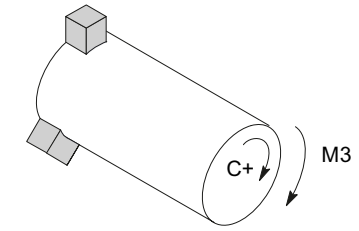
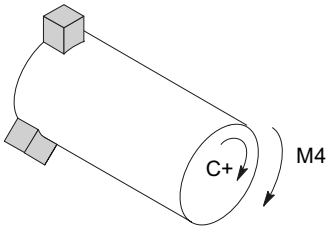
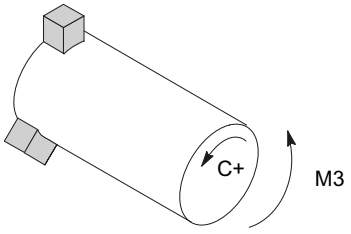
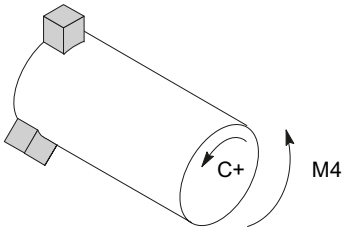
## Defining the direction of rotation

In order to ensure that the rotational direction of the spindle and C axis is correctly displayed on the user interface and when programming, the correct direction of rotation is executed, you must make several settings that are coordinated with one another. You must base these settings on the actual direction of rotation of the spindle/C axis on the machine.

You specify the settings in the following machine data:

| MD52207 $MCS_AXIS_USAGE_ATTRIB[n] | Attributes of the axes<br>[n] channel axis number |
|---|---|
| Bit 3 | Displayed positive direction of rotation is counter-clockwise (for rotary axes)<br>The view must be from the inside of the machine (operator's view) to the axis of rotation. |
| Bit 4 | Displayed direction of rotation for M3 is counter-clockwise (for spindles)<br>This bit can be set from the operator's view or according to DIN 66025. |
| Bit 5 | Direction of rotation for M3 corresponds to minus rotary axis (for spindles)<br>You must set this bit as for PLC bit DB38xx.DBX2001.6. |

As a result, the following setting options are obtained for the main spindle:

| Main spindle direction of rotation | 52207[n] bit 3 = | 52207[n] bit 4 = *) | 52207[n] bit 5 = | DB38xx DBX2001.6 |
|---|---|---|---|---|
|  | 0 | 0 | 0 | 0 |
|  | 0 | 1 | 1 | 1 |
|  | 1 | 1 | 0 | 0 |
|  | 1 | 0 | 1 | 1 |

**Note**

**MD52207[n] bit 4 = *)**

Bit 4 can be set from the operator's view (values as specified) or according to DIN ("0" and "1" are then interchanged).

## Dimensions

To define the reference point for moving the counterspindle, you must first communicate the dimensions of the counterspindle. You can either enter the dimensions in the following axis-specific cycle machine data or in the menu "Parameter" -> Setting data -> Spindle chuck data" Changes to the machine data are automatically accepted in the menu and vice versa.

| MD53240 $MAS_SPINDLE_PARAMETER[ ] | | Spindle chuck data |
|---|---|---|
| [0] | Chuck dimensions | |

| | |
|---|---|
| [1] | Stop dimensions |
| [2] | Jaw dimensions |

| MD53241 $MAS_SPINDLE_CHUCK_TYPE[ ] | Spindle jaw type |
|---|---|
| = 0 | Clamping, outer |
| = 1 | Clamping, inner |

## Additional settings

The M code, e.g. M34 or M1 = 34 for the spindle chuck is defined in the following machine data. The manufacturer cycle CUST_TECHCYC.SPF also takes the M functions from the following machine data:

| MD52250 $MCS_M_CODE_CHUCK_OPEN[ ] | M code for open chuck with stationary spindle |
|---|---|
| [0] | Main spindle |
| [1] | Counterspindle |

| MD52251 $MCS_M_CODE_CHUCK_CLOSE_OPEN_ROT[ ] | M code for open chuck with spindle rotating |
|---|---|
| [0] | Main spindle |
| [1] | Counterspindle |

| MD52252 $MCS_M_CODE_CHUCK_CLOSE[ ] | M code for close chuck |
|---|---|
| [0] | Main spindle |
| [1] | Counterspindle |

| MD52253 $MCS_M_CODE_TAILSTOCK_FORWARD[ ] | M code for quill forward |
|---|---|
| [0] | Quill opposite the main spindle |
| [1] | |

| MD52254 $MCS_M_TAILSTOCK_BACKWARD[ ] | M code for quill back |
|---|---|
| [0] | Quill opposite the main spindle |
| [1] | Quill opposite the main spindle |

### Rounding the contour

| SD55582 $SCS_TURN_CONT_TRACE_ANGLE | Contour turning: Minimum angle for rounding on the contour |
|---|---|
| = 5 | Specifies the angle between the cutting edge and contour, above which for contour turning, the contour is rounded in order to remove residual material (default value). |

## 7.5.3 Setting-up ShopTurn cycles for turning

| MD52210 $MCS_FUNCTION_MASK_DISP | Function mask display |
|---|---|
| Bit 0 = 1 | Dimension system for programs always in the basic system (default setting) |
| Bit 1 = 1 | Face view when turning (default setting) |
| Bit 4 = 1 | Display follow-on tool in the T,F,S window |

| MD52212 $MCS_FUNCTION_MASK_TECH | Cross-technology function mask |
|---|---|
| Bit 2 | Approach logic for stepped drill |
| Bit 3 | Block search cycle for ShopMill / ShopTurn |
| = 0 | The E__ASUP or F_S_ASUP cycles are not called in the block search cycle PROG_EVENT.SPF |
| = 1 | The E__ASUP (under ShopMill) or F_S_ASUP (under ShopTurn) cycles are not called in the block search cycle PROG_EVENT.SPF |
| Bit 4 | Approach logic using the cycle |
| Bit 5 | The cycle for SERUPRO (CYCLE207) is called in the block search cycle PROG_EVENT.SPF. |
| Bit 6 | Work offset value WO cannot be entered |
| Bit 7 | Identification of the end of tool lives in the program (ShopMill / ShopTurn) |
| Bit 8 | Manual machine (ShopMill/ShopTurn) |
| Bit 9 | Selection/deselection of the work offset using softkey |
| Bit 11 | Deactivate position check for drilling and milling tools |
| Bit 12 | Alarm at positive Z offset |

| MD52214 $MCS_FUNCTION_MASK_MILL | Milling function screen |
|---|---|
| Bit 3 | "Inner/rear" machining is enabled in the ShopTurn screens which themselves define the machining plane. |
| Bit 4 | If you have implemented the "Clamp/release spindle" function using the manufacturer cycle CUST_TECHCYC.SPF, then using this machine data, you can activate the "Clamp/release spindle" parameter in the drilling and milling screens. |
| = 0 | The "Clamp/release spindle" parameter is not displayed in the drilling and milling screens. ShopTurn automatically clamps the spindle if it makes sense for the particular machining operation. |
| = 1 | The "Clamp/release spindle" parameter is displayed in the drilling and milling screens. You decide for which machining operation the spindle should be clamped. |

| MD52216 $MCS_FUNCTION_MASK_DRILL | Drilling function screen form |
|---|---|
| Bit 3 | Constant cutting speed referred to the diameter of the centering |
| = 0 | Constant cutting speed referred to the diameter of the tool |
| = 1 | Constant cutting speed referred to the diameter of the centering |

| MD52218 $MCS_FUNCTION_MASK_TURN | Turning function screen |
|---|---|
| Bit 2 | Enable tailstock |

| | |
|---|---|
| Bit 5 | Enable spindle control of the tool spindle via user interface |
| Bit 6 | Enable Balance Cutting for two-channel stock removal |

| MD52229 $MCS_ENABLE_QUICK_M_CODES | Enable fast M functions |
|---|---|
| Bit 0 | Coolant OFF |
| Bit 1 | Coolant 1 ON |
| Bit 2 | Coolant 2 ON |
| Bit 3 | Coolants 1 and 2 ON |

| MD52230 $MCS_M_CODE_ALL_COOLANTS_OFF | M code for all coolants OFF |
|---|---|
| = 9 | This machine data is used to define the M function for switching off the coolant; which is output when the tool is changed. |

| MD52231 $MCS_M_CODE_COOLANT_1_ON | M code for coolant 1ON |
|---|---|
| = 8 | This machine data is used to define the M function for coolant 1, which is output when the tool is changed. |

| MD52232 $MCS_M_CODE_COOLANT_2_ON | M code for coolant 2ON |
|---|---|
| = 7 | This machine data is used to define the M function for coolant 2, which is output when the tool is changed. |

| MD52233 $MCS_M_CODE_COOLANT_1_AND_2_ON | M code for both coolants ON |
|---|---|
| = -1 | This machine data is used to define the M function for coolant 1 and 2, which is output when the tool is changed. |

| SD55505 $SCS_TURN_ROUGH_O_RELEASE_DIST | Retraction distance for stock removal during external machining |
|---|---|
| = 1 | Specifies the distance, by which the tool is retracted from the contour when removing stock from an outer corner. This does not apply to stock removal at a contour (default value). |
| = -1 | The distance is internally defined. |

| SD55506 $SCS_TURN_ROUGH_I_RELEASE_DIST | Retraction distance for stock removal during internal machining |
|---|---|
| = 0.5 | Specifies the distance, by which the tool is retracted from the contour when removing stock from an internal corner. This does not apply to stock removal at a contour (default value). |
| = -1 | The distance is internally defined. |

## 7.5.4 Setting up a counterspindle



**Software option**

You require the following software option in order to use the counterspindle on your machine:

"Generic coupling 'CP-STATIC'" or higher

### Function

If your turning machine has a counterspindle, you can machine workpieces using turning, drilling and milling functions on the front and rear faces without reclamping the workpiece manually.

Before machining the rear face, the counterspindle must grip the workpiece, pull it out of the main spindle, and position it at the new machining position.

### Meaning of the axes

| MD52206 $MCS_AXIS_USAGE[n] | Meaning of the axes in the channel<br>[n] channel axis number |
|---|---|

A detailed description of MD52006 is provided in Chapter "Activating turning/milling/drilling/grinding technologies (Page 71)".

### Traversing direction

In order that the user interface and cycles know in which direction the spindles and rotary axes rotate, the following settings must be made.

You set the direction of rotation of an NC rotary axis via MD32100 $MA_AX_MOTION_DIR.

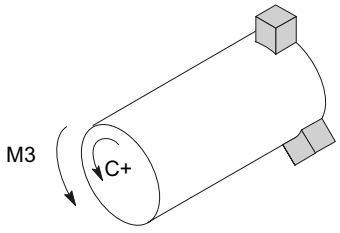
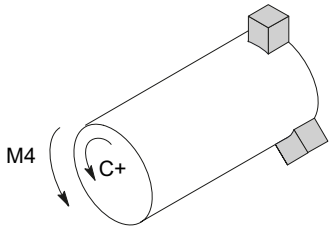
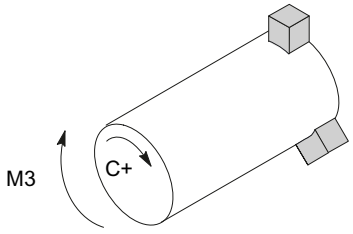
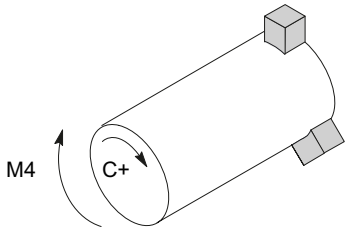
| MD32100 $MA_AX_MOTION_DIR | Traversing direction (not control direction) |
|---|---|
| -1 | Direction reversal |
| 0, 1 | No direction reversal |

### Defining the direction of rotation

You set the directions of rotation for the user interface in the following machine data:

| MD52207 $MCS_AXIS_USAGE_ATTRIB[n] | Attributes of the axes in the channel<br>[n] channel axis number |
|---|---|
| Bit 3 | Displayed positive direction of rotation is counter-clockwise (for rotary axes)<br>The view must be from the inside of the machine (operator's view) to the axis of rotation. |
| Bit 4 | Displayed direction of rotation for M3 is counter-clockwise (for spindles)<br>This bit can be set from the operator's view or according to DIN. |
| Bit 5 | Direction of rotation for M3 corresponds to minus rotary axis (for spindles)<br>You must set this bit as for PLC bit DB38xx.DBX2001.6. |

As a result, the following setting options are obtained for the counterspindle:

| Counterspindle direction of rotation | 52207[n] bit 3 = | 52207[n] bit 4 = *) | 52207[n] bit 5 = | DB38xx. DBX2001.6 |
|---|---|---|---|---|
|  | 1 | 1 | 0 | 0 |
|  | 1 | 0 | 1 | 1 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 1 | 1 | 1 |

**Note**

**MD52207[n] bit 4 = *)**

Bit 4 can be set from the operator's view (values as specified) or according to DIN ("0" and "1" are then interchanged).

The display must be the same for the main spindle and the counterspindle ("from operator's view" or "according to DIN").

### Dimensions

To define the reference point for moving the counterspindle, you must first announce the dimensions of the counterspindle. You can either enter the dimensions in the following axis-specific cycle machine data or in the menu "Parameter". → "Setting data" → "Spindle chuck data". Changes to the machine data are automatically accepted in the menu and vice versa.

| MD53240 $MAS_SPINDLE_PARAMETER[ ] | Spindle chuck data |
|---|---|
| [0] | Chuck dimensions |
| [1] | Stop dimensions |
| [2] | Jaw dimensions |

| MD52241 $MCS_SPINDLE_CHUCK_TYPE | Spindle jaw type |
|---|---|
| = 0 | Clamping, outer |
| = 1 | Clamping, inner |

## 7.5.5 Setting up the counterspindle under ShopTurn



**Software option**

You require the following software option in order to use the counterspindle on your machine:

"Generic coupling 'CP-STATIC'" or higher

The position to which the counterspindle travels when the program starts is defined in the following channel-specific cycle setting data:

| SD55232 $SCS_SUB_SPINDLE_REL_POS | Retraction position Z for counterspindle |
|---|---|

The following channel-specific setting data become effective when traveling to the fixed stop:

| SD55550 $SCS_TURN_FIXED_STOP_DIST | Distance for travel to fixed stop. |
|---|---|
| SD55551 $SCS_TURN_FIXED_STOP_FEED | Feedrate for travel to fixed stop. |
| SD55552 $SCS_TURN_FIXED_STOP_FORCE | Force for travel to fixed stop in %. |

Between traveling to the fixed stop and gripping, the counterspindle can retract a short distance to counteract compressive stress in the workpiece.

| SD55553 $SCS_TURN_FIXED_STOP_RETRACTION | Retraction distance before clamping after fixed stop. |
|---|---|

After gripping you can cut off the workpiece. Before doing so, the counterspindle can retract a short distance with the workpiece to exert tensile stress on the workpiece. This relieves pressure on the tool when cutting off.

| SD55543 $SCS_TURN_PART_OFF_RETRACTION | Retraction distance before cut-off. |
|---|---|

After parting, you can carry out a cut-off check and for turning, use the "Travel to fixed stop" function. You can activate/deactivate the cut-off check using the channel-specific cycle setting data:

| | |
|---|---|
| SD55540 $SCS_TURN_PART_OFF_CTRL_DIST | Distance for cut-off check. |
| SD55541 $SCS_TURN_PART_OFF_CTRL_FEED | Feedrate for cut-off check. |
| SD55542 $SCS_TURN_PART_OFF_CTRL_FORCE | Force for cut-off check in %. |

The cut-off is successful when travel to fixed stop fails. The following alarms are output:

| Alarm | Alarm text |
|---|---|
| 20091 | Axis %1 has not reached the fixed stop. |
| 20094 | Axis %1 end stop has been aborted. |

You can switch off the alarm display using the following machine data:

| | |
|---|---|
| MD37050 $MA_FIXED_STOP_ALARM_MASK | Enabling the fixed stop alarms. |
| = 2 | Suppressing alarms 20091 and 20094. |

You can set this machine data axis-specifically in the "Machine Data" window in the "Tool zero" operating area.

If, however, the specified force is reached during the cut-off check (i.e. travel to fixed stop is successful), alarm 61255 "Error during cut-off: Tool break?" is issued.

**Note**

The "Travel to fixed stop" function can also be used when gripping the spindle (see above). If travel to fixed stop does not succeed when gripping, an alarm will of course also be issued. Instead of alarms 20091 and 20094, the alarm 61254 "Error during travel to fixed stop" will be issued.

### Machine manufacturer cycle

If you perform one of the following actions, you must adapt the machine manufacturer cycle CUST_TECHCYC.SPF.

- Switch between spindle and C axis mode for the main spindle or counterspindle.
- Open, close, flush chuck (main spindle / counterspindle).
- Change default settings for the coupling of the main spindle and counterspindle.

## 7.5.6 Technology cycles for turning

### Thread-cutting (CYCLE99)

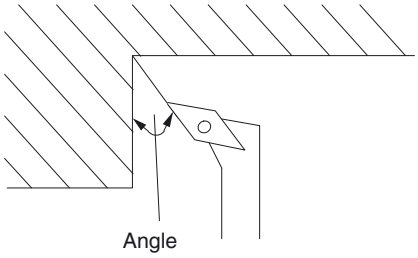
To correctly machine the thread, it is necessary to set the machine data MD52207 $MCS_AXIS_USAGE_ATTRIB[n] with bit 3, see Chapter: Setting up a counterspindle (Page 107).

## Contour groovingCYCLE930, contour turning CYCLE950, stock removal at corner CYCLE951, contour turning CYCLE952

| SD55500 $SCS_TURN_FIN_FEED_PERCENT | | Enter the finishing feedrate for complete machining, roughing and finishing. The percentage of the value corresponds to that entered under parameter F (feedrate). |
|---|---|---|
| = 100 | 100% finishing feedrate | |

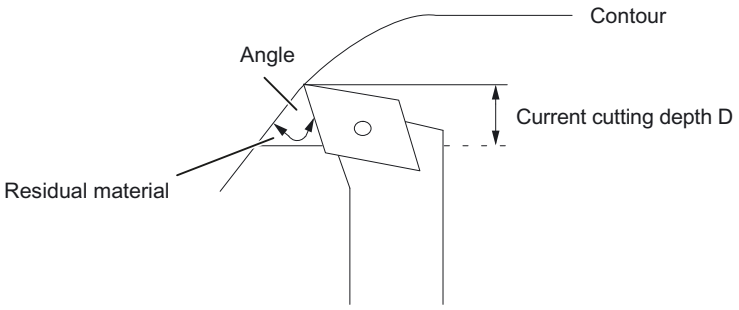
| SD55510 $SCS_TURN_GROOVE_DWELL_TIME | | Dwell time, which is necessary between grooving and retracting for grooving technology. Tool clearance time during grooving at the base. |
|---|---|---|
| = > 0 | Dwell time in seconds | |
| = < 0 | Dwell time in spindle revolutions | |

| SD55580 $SCS_TURN_CONT_RELEASE_ANGLE | | Angle, through which the tool is lifted from the contour for contour turning, roughing. |
|---|---|---|
| = 45 | Retraction angle of 45 degrees | |




| SD55581 $SCS_TURN_CONT_RELEASE_DIST | | Amount, by which the tool is lifted when roughing a contour taking into account the retraction angle SD55580 $SCS_TURN_CONT_RELEASE_ANGLE.<br>This also applies to stock removal, grooving and plunge turning. |
|---|---|---|
| = 1 | 1 mm or 1 inch retraction distance | |

| SD55582 $SCS_TURN_CONT_TRACE_ANGLE | | The angle between cutting edge and contour as of which rounding is performed on the contour during contour turning in order to remove residual material.<br>If the angle of the residual material is greater than that specified in the setting data, the tool will round the contour. |
|---|---|---|
| = 5 | 5 degree angle | |




| SD55583 $SCS_TURN_CONT_VARIABLE_DEPTH | Percentage of the changing cutting depth when turning a contour<br>You can select changing the cutting depth for stock removal and for removing residual material. |
|---|---|
| = 20 | 20% variable cutting depth |

| SD55584 $SCS_TURN_CONT_BLANK_OFFSET | Safety clearance when avoiding obstacles, to avoid collisions during approach or retraction from the machining.<br>This also applies to stock removal, grooving and plunge turning. |
|---|---|
| = 1 | 1 mm or 1 inch blank allowance |

| SD55585 $SCS_TURN_CONT_INTERRUPT_TIME | Time for the feedrate interruption for contour turning.<br>This also applies to stock removal, grooving and plunge turning. |
|---|---|
| = > 0 | Interruption time in seconds |
| = < 0 | Interruption time in revolutions |
| = 0 | No interruption |

**Note**

The channel-specific cycle setting data SD55585 is only evaluated if SD55586 $SCS_TURN_CONT_INTER_RETRACTION = 0.

| SD55586 $SCS_TURN_CONT_INTER_RETRACTION | Retraction distance for contour turning for feedrate interruption.<br>This also applies to stock removal, grooving and plunge turning. |
|---|---|
| = > 0 | Retraction distance at feed interruption<br>SD55585 $SCS_TURN_CONT_INTERRUPT_TIME has no effect. |
| = 0 | No retraction distance |

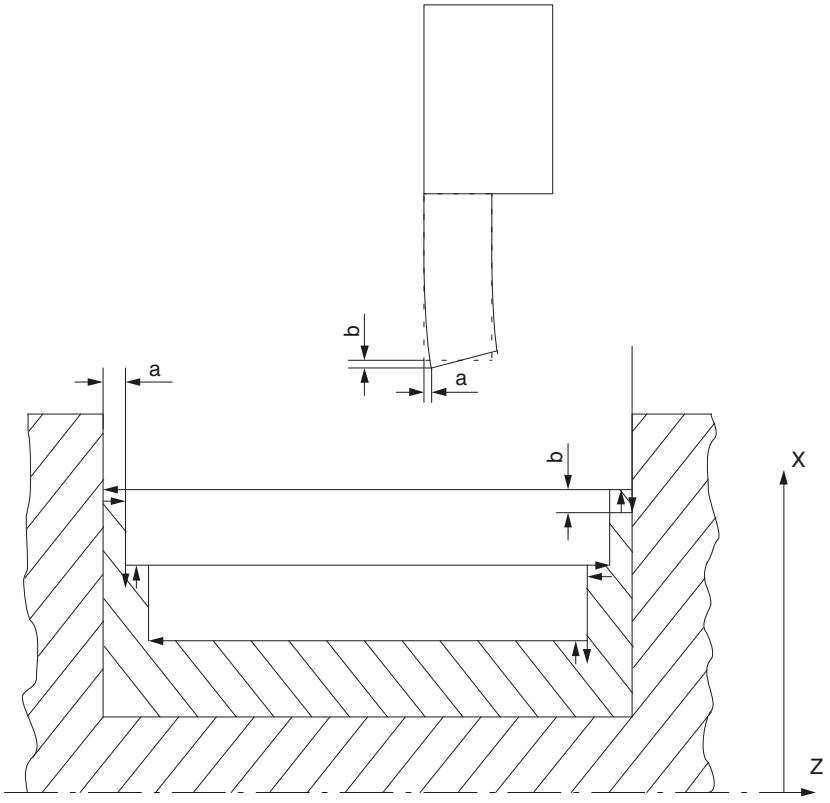
| SD55587 $SCS_TURN_CONT_MIN_REST_MAT_AX1 | | Limit value during contour turning for removal of residual material in the direction of axis 1 (for G18 Z).<br>This also applies to residual machining – stock removal, grooving and plunge turning. |
|---|---|---|
| 50 | The roughing step for residual machining is not executed, if the material to be removed is less than 50% of the finishing allowance. The material is only removed when finishing. | |

Example: If the SD is set to 50% and the final machining allowance is 0.5 mm, any residual material less than 0.25 mm is not removed with the residual machining – but is removed during finishing. If during a machining step, less residual material is present than defined in the SD, the error message "No material present" is issued. This means that this residual material roughing step can be omitted because no machining is performed.

| SD55588 $SCS_TURN_CONT_MIN_REST_MAT_AX2 | | Limit value during contour turning for removal of residual material in the direction of axis 2 (for G18 X).<br>This also applies to residual machining – stock removal, grooving and plunge turning. |
|---|---|---|
| 50 | The roughing step for residual machining is not executed, if the material to be removed is less than 50% of the finishing allowance. The material is only removed when finishing. | |

As the tool bends during plunge turning, the tool cannot travel right up to the contour during stock removal. The lateral distance to the last cut by which the next cut is shortened is specified in the following channel-specific cycle setting data.

| SD55595 $SCS_TURN_CONT_TOOL_BEND_RETR | | Retraction distance because of tool bending |
|---|---|---|
| 0.1 | 0.1 mm or 0.1 inch retraction distance | |




a Distance to the last cut, SD55595

b Retraction between plunge-cutting and stock removal, SD55596

As the tool bends during plunge turning, the tool would make an excessively deep cut during stock removal. The retraction distance of the tool between plunge-cutting and stock removal is specified in the following channel-specific cycle setting data:

| SD55596 $SCS_TURN_CONT_TURN_RETRACTION | | Retraction depth before turning |
|---|---|---|
| = 0.1 | 0.1 mm or 0.1 inch retraction depth | |

## 7.5.7 Axis configuration of a lathe

### General configuration

If driven milling tools are available on a lathe, then the following functions can also be set-up on this machine:

- Cylinder surface transformation (TRACYL) (Page 115)
- End face machining (TRANSMIT) (Page 119)

### Lathe with X1 and Z1 axis, main spindle and tool spindle

You can configure the following channel-specific machine data, for example, for a lathe with X1 and Z1 axis as well as a main spindle (C1) and a tool spindle (SP1):

| MD20080 $MC_AXCONF_CHANAX_NAME_TAB[ ] | Channel axis name in the channel |
|---|---|
| [0] = X1 | Channel axis X1 |
| [1] = Z1 | Channel axis Z1 |
| [2] = C1 | Main spindle C1 |
| [3] = SP1 | Tool spindle SP1 |

### Lathe with X1 and Z1 axis, main spindle and tool spindle and counterspindle

You can configure the following machine data, for example, for a lathe with X1 and Z1 axis as well as a main spindle (C1), a tool spindle (SP1) and a counterspindle (C2):

| MD20080 $MC_AXCONF_CHANAX_NAME_TAB[ ] | Channel axis name in the channel |
|---|---|
| [0] = X1 | Channel axis X1 |
| [1] = Z1 | Channel axis Z1 |
| [2] = C1 | Main spindle C1 |
| [3] = SP1 | Tool spindle SP1 |
| [5] = C2 | Counterspindle C2 |

### Lathe with X1 and Z1 axis, main spindle and tool spindle and Y1 axis

You can configure the following machine data, for example, for a lathe with X1, Z1 and Y1 axis as well as a main spindle (C1) and a tool spindle (SP1):

| MD20080 $MC_AXCONF_CHANAX_NAME_TAB[ ] | Channel axis name in the channel |
|---|---|
| [0] = X1 | Channel axis X1 |
| [1] = Z1 | Channel axis Z1 |
| [2] = C1 | Main spindle C1 |
| [3] = SP1 | Tool spindle WSP1 |
| [5] = Y1 | Channel axis Y1 |

## 7.5.8 Cylinder surface transformation (TRACYL)



**Software option**

You require the following software option in order to use this function:

"Transmit and peripheral surface transformation"

### Function

Using the cylinder surface transformation function (TRACYL), you can machine the peripheral surface of a turned part.

## Setting up

General settings for cylinder transformation

| MD10602 $MN_FRAME_GEOAX_CHANGE_MODE | | Frames when switching over geometry axes. |
|---|---|---|
| = 1 | The actual total frame (work offsets) and when switching over geometry axes (selecting - deselecting TRACYL) is recalculated. | |

| MD24040 $MC_FRAME_ADAPT_MODE | Adapting the active frames |
|---|---|
| Bit 0 = 1 | |
| Bit 1 = 1 | |
| Bit 2 = 1 | |

| MD28082 $MC_MM_SYSTEM_FRAME_MASK | System frame (SRAM) |
|---|---|
| Default, cannot be changed | |

When setting up the functions, you can take the following channel-specific machine data into account:

| MD24300 $MC_TRAFO_TYPE_3 | | Cylinder surface transformation, main spindle: Transformation 3 |
|---|---|---|
| = 512 | without groove wall correction (without Y1 axis) | |
| = 513 | with groove wall correction (with Y1 axis) | |
| = 514 | with groove wall correction and Y1 correction | |

| MD24400 $MC_TRAFO_TYPE_4 | | Cylinder surface transformation, counterspindle: Transformation 4 |
|---|---|---|
| = 512 | without groove wall correction (without Y1 axis) | |
| = 513 | with groove wall correction (with Y1 axis) | |
| = 514 | with groove wall correction and Y1 correction | |

**Note**

You must also set up other machine data for each of the individual transformations.

## Setting up cylinder surface transformation under ShopTurn

In addition, set the following channel-specific configuration machine data:

| MD52214 $MC_FUNCTION_MASK_MILL | Milling function screen |
|---|---|
| Bit 3 | Enable "inside/rear" machining in ShopTurn screens which define the machining plane themselves. |
| Bit 4 | If you have implemented the "Clamp/release spindle" function by means of the machine manufacturer cycle CUST_TECHCYC.SPF, you can activate the "Clamp/release spindle" parameter in the drilling and milling screens with this machine data. |
| = 0 | The "Clamp/release spindle" parameter is not displayed in the drilling and milling masks.<br>ShopTurn automatically clamps the spindle if it makes sense for the particular machining operation. |
| =1 | The "Clamp/release spindle" parameter is displayed in the drilling and milling masks.<br>The operator decides for which machining operation the spindle should be clamped. |

## Cylinder surface transformation without groove side offset

| MD24300 $MC_TRAFO_TYPE_3 | Definition of the 3rd transformation in the channel |
|---|---|
| = 512 | TRACYL main spindle. |

| MD24310 $MC_TRAFO_AXES_IN_3[ ] | Axis assignment for transformation 3 |
|---|---|
| [0] = 1 | Perpendicular to the rotary axis X1 |
| [1] = 3 | Rotary axis (main spindle) C1 |
| [2] = 2 | Parallel to the rotary axis Z1 |

| MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[ ] | Assignment of the geometry axes to channel axes for transformation 3. |
|---|---|
| [0] = 1 | 1. Channel axis X1 |
| [1] = 3 | 2. Channel axis Y1 |
| [2] = 2 | 3. Channel axis Z1 |

| MD24800 $MC_TRACYL_ROT_AX_OFFSET_1 | Offset of the rotary axis for the 1st TRACYL transformation. |
|---|---|
| = 0 | |

| MD24805 $MC_TRACYL_ROT_AX_FRAME_1 | Axial offset of the rotary axis is taken into account during TRACYL. |
|---|---|
| = 2 | |

| MD24810 $MC_TRACYL_ROT_SIGN_IS_PLUS_1 | Sign of the rotary axis for the 1st TRACYL transformation. |
|---|---|
| = 1 | |

| MD24820 $MC_TRACYL_BASE_TOOL_1[ ] | Vector of the basis tool for the 1st TRACYL transformation. |
|---|---|
| [0] = 0 | |
| [1] = 0 | |
| [2] = 0 | |

## Cylinder surface transformation with groove side offset

| MD24300 $MC_TRAFO_TYPE_3 | Definition of the 3rd transformation in the channel |
|---|---|
| = 513 | TRACYL main spindle |

| MD24310 $MC_TRAFO_AXES_IN_3[ ] | Axis assignment for transformation 3 |
|---|---|
| [0] = 1 | Perpendicular to the rotary axis X1 |
| [1] = 3 | Rotary axis (main spindle) C1 |
| [2] = 2 | Parallel to the rotary axis Z1 |
| [3] = 6 | parallel to the cylinder surface and perpendicular to the rotary axis Z1 |

| MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[ ] | Assignment of the geometry axes to channel axes for transformation 3. |
|---|---|
| [0] = 1 | 1. Channel axis X1 |
| [1] = 3 | 2. Channel axis Y1 |
| [2] = 2 | 3. Channel axis Z1 |

| MD24800 $MC_TRACYL_ROT_AX_OFFSET_1 | Offset of the rotary axis for the 1st TRACYL transformation. |
|---|---|
| = 0 | |

| MD24805 $MC_TRACYL_ROT_AX_FRAME_1 | Axial offset of the rotary axis is taken into account during TRACYL. |
|---|---|
| = 2 | |

| MD24810 $MC_TRACYL_ROT_SIGN_IS_PLUS_1 | Sign of the rotary axis for the 1st TRACYL transformation. |
|---|---|
| = 1 | |

| MD24820 $MC_TRACYL_BASE_TOOL_1[ ] | Vector of the basis tool for the 1st TRACYL transformation. |
|---|---|
| [0] = 0 | |
| [1] = 0 | |
| [2] = 0 | |

## 7.5.9 End face machining (TRANSMIT)



**Software option**
You require the following software option in order to use this function:
"Transmit and peripheral surface transformation"

### Function

The end faces of a turned part are machined with the end face machining function (TRANSMIT).

### Setting up

For general settings for transformations, please refer to Chapter "Cylinder surface transformation (TRACYL) (Page 115)".

You can make additional settings in the following channel-specific machine data:

| MD24100 $MC_TRAFO_TYPE_1 | | End face machining, main spindle: Transformation 1 |
|---|---|---|
| = 256 | Machining without Y1 axis | |
| = 257 | Machining with Y1 axis | |

| MD24200 $MC_TRAFO_TYPE_2 | | End face machining, counterspindle: Transformation 2 |
|---|---|---|
| = 256 | Machining without Y1 axis | |
| = 257 | Machining with Y1 axis | |

| MD24110 $MC_TRAFO_AXES_IN_1[ ] | | Axis assignment for the 1st transformation in the channel |
|---|---|---|
| [0] = 1 | Perpendicular to the rotary axis X1 | |
| [1] = 3 | Rotary axis (main spindle) C1 | |
| [2] = 2 | Parallel to the rotary axis Z1 | |

| MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[ ] | | Assignment of the geometry axes to channel axes for transformation 1 |
|---|---|---|
| [0] = 1 | 1. Channel axis X1 | |
| [1] = 3 | 2. Channel axis Y1 | |
| [2] = 2 | 3. Channel axis Z1 | |

| MD24900 $MC_TRANSMIT_ROT_AX_OFFSET_1 | | |
|---|---|---|
| = 0 | Offset of the rotary axis for the 1st TRANSMIT transformation. | |

| MD24905 $MC_TRANSMIT_ROT_AX_FRAME_1 | |
|---|---|
| = 2 | Axial offset of the rotary axis is taken into account during TRANSMIT 1. |

| MD24910 $MC_TRANSMIT_ROT_SIGN_IS_PLUS_1 | |
|---|---|
| = 0 | Sign of the rotary axis for the 1st TRANSMIT transformation. |

| MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 | |
|---|---|
| = 1 | Limitation of working range in front of/behind the pole, 1st TRANSMIT. |

| MD24920 $MC_TRANSMIT_BASE_TOOL_1[ ] | Vector of the basis tool for the 1st TRANSMIT transformation. |
|---|---|
| [0] = 0 | |
| [1] = 0 | |
| [2] = 0 | |

## TRANSMIT with real Y axis

| MD24100 $MC_TRAFO_TYPE_1 | |
|---|---|
| = 257 | Definition of transformation 1 in the channel: TRANSMIT main spindle. |

| MD24110 $MC_TRAFO_AXES_IN_1[ ] | Axis assignment for the 1st transformation in the channel. |
|---|---|
| [0] = 1 | Perpendicular to the rotary axis X1 |
| [1] = 3 | Rotary axis C1 |
| [2] = 2 | Parallel to the rotary axis Z1 |

| MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[ ] | Assignment of the geometry axes to channel axes for transformation 1. |
|---|---|
| [0] = 1 | 1. Channel axis X1 |
| [1] = 3 | 2. Channel axis Y1 |
| [2] = 2 | 3. Channel axis Z1 |

| MD24900 $MC_TRANSMIT_ROT_AX_OFFSET_1 | |
|---|---|
| = 0 | Offset of the rotary axis for the 1st TRANSMIT transformation. |

| MD24905 $MC_TRANSMIT_ROT_AX_FRAME_1 | |
|---|---|
| = 2 | Axial offset of the rotary axis is taken into account during TRANSMIT 1. |

| MD24910 $MC_TRANSMIT_ROT_SIGN_IS_PLUS_1 | |
|---|---|
| = 0 | Sign of the rotary axis for the 1st TRANSMIT transformation. |

| MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 | |
|---|---|
| = 1 | Limitation of working range in front of/behind the pole, 1st TRANSMIT. |

| MD24920 $MC_TRANSMIT_BASE_TOOL_1[ ] | Vector of the basis tool for the 1st TRANSMIT transformation. |
|---|---|
| [0] = 0 | |
| [1] = 0 | |
| [2] = 0 | |

## Face machining under ShopTurn

If you wish to use the face machining function under ShopTurn, set the following additional channel-specific configuration machine data:

| MD52214 $MCS_FUNCTION_MASK_MILL | ShopTurn function screen |
|---|---|
| Bit 3 | Enable "internal / rear" machining in ShopTurn screens that define the machining plane themselves. |
| Bit 4 | If you have implemented the "Clamp/release spindle" function using the machine manufacturer cycle CUST_TECHCYC.SPF, then using this machine data, you can activate the "Clamp/release spindle" parameter in the drilling and milling masks. |
| = 0 | The "Clamp/release spindle" parameter is not displayed in the drilling and milling masks. ShopTurn automatically clamps the spindle if it makes sense for the particular machining operation. |
| = 1 | The "Clamp/release spindle" parameter is displayed in the drilling and milling screens. The operator decides for which machining operation the spindle should be clamped. |

**Note**

The face end machining is automatically integrated in the cycles, with the exception of the straight line and circle.

You can select the functions for these two cycles in the "Program" operating area at "Straight line" and "Circle".

## References

Additional information on face end machining is provided in:

Function Manual, Extended Functions; Kinematic Transformation (M1): TRANSMIT

## 7.5.10 Inclined Y axis (TRAANG)



**Software option**

You require the following software option in order to use the inclined Y axis function on your machine:

"Inclined axis" or "Inclined axis, fixed angle"

### Function

If your lathe has an inclined Y axis (i.e. this axis is not perpendicular to axes X and Z), you can still completely program machining operations in Cartesian coordinates. The control uses the "Inclined axis function (TRAANG)" to transform the Cartesian coordinates to the motion of the inclined axis.

### Setting up

You still have to set up the inclined axis function (TRAANG) via machine data.

### Inclined axis under ShopTurn

Please also note the following:

**Note**

In the user interface, the inclined axis function is automatically integrated in the cycles after setting up. You can select the machining plane "Face Y1" or "Surface Y1" for machining with inclined axis in the screens and enter Cartesian coordinates.

### References

Function Manual, Extended Functions; Kinematic Transformations (M1): TRAANG

### Example

You must configure the following machine data, for example, for a lathe with X1, Z1 and Y1 axis as well as a main spindle (C1) and a tool spindle (SP1):

| MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[ ] | | Assignment of geometry axis to channel axis. |
|---|---|---|
| [0] = 1 | 1. real geometry axis X1 axis. | |
| [1] = 0 | 2. real geometry axis Y1 axis not available. | |
| [2] = 2 | 3. real geometry axis Z1 axis. | |

| MD20110 $MC_RESET_MODE_MASK | |
|---|---|
| Bit 0 = 1<br>Bit 7 = 0 | TRAANG is retained after ramp-up. |

| MD20112 $MC_START_MODE_MASK | |
|---|---|
| Bit 7 = 1 | TRAANG is retained after "Cycle start". |

| MD20118 $MC_GEOAX_CHANGE_RESET | |
|---|---|
| = 1 | Allow automatic geometry axis change. |

| MD20140 $MC_TRAFO_RESET_VALUE | |
|---|---|
| = 5 | TRAANG always active after reset. |

| MD20144 $MC_TRAFO_MODE_MASK | |
|---|---|
| Bit 0 = 1 | TRAANG runs in the background (persistent) and is not shown on the user interface. |

| MD20070 $MC_AXCONF_MACHAX_USED[4] | |
|---|---|
| = 5 | Channel axis Y1 = 5th machine axis. |

| MD20080 $MC_AXCONF_CHANAX_NAME_TAB[ ] | Name of channel axis in the channel. |
|---|---|
| [0] = X1 | 1. Channel axis X1 |
| [1] = Z1 | 2. Channel axis Z1 |
| [2] = C1 | 3. Channel axis C1 |
| [3] = SP1 | 4. Channel axis SP1 |
| [4] = Y1 | 5. Channel axis Y1 |

### Data set for inclined axis:

| MD24430 $MC_TRAFO_TYPE_5 | |
|---|---|
| = 1024 | Transformation 5: TRAANG |

| MD24432 $MC_TRAFO_AXES_IN_5[ ] | Axis assignment for transformation 5. |
|---|---|
| [0] = 5 | 1. Transformation axis = channel axis, Y1 |
| [1] = 1 | 2. Transformation axis = channel axis X1 |
| [2] = 2 | 3. Transformation axis = channel axis Z1 |

| MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[ ] | Assignment of geometry axes to channel axes for transformation 5. |
|---|---|
| [0] = 1 | 1. Axis = channel axis X1 |
| [1] = 5 | 2. Axis = channel axis Y1 |
| [2] = 2 | 3. Axis = channel axis Z1 |

| MD24436 $MC_TRAFO_INCLUDES_TOOL_5 | |
|---|---|
| = 0 | Tool handling with active transformation 5. |

| MD24700 $MC_TRAANG_ANGLE_1 | |
|---|---|
| = 55 | Angle between 1st and 2nd transformer axis. Data record for linking (TRACON) of end face machining on main spindle (TRANSMIT) and inclined axis (TRAANG). |

### Data record for linking (TRACON) of end face machining on main spindle (TRANSMIT) and inclined axis (TRAANG):

| MD24440 $MC_TRAFO_TYPE_6 | |
|---|---|
| = 8192 | Type of transformation that is available as sixth in the channel. |

| MD24444 $MC_TRAFO_GEOAX_ASSIGN_TAB_6[ ] | Assignment of geometry axes to channel axes for transformation 6. |
|---|---|
| [0] = 1 | 1. Axis = channel axis X1 |
| [1] = 3 | 2. Axis = channel axis Y1 |
| [2] = 2 | 3. Axis = channel axis Z1 |

| MD24995 $MC_TRACON_CHAIN_1[ ] | Transformation linking |
|---|---|
| [0] = 1 | Number of the TRANSMIT transformation (main spindle) for linking. |
| [1] = 5 | Number of the TRAANG transformation for linking. Data set for linking (TRACON) of cylinder surface transformation on main spindle (TRACYL) and inclined axis (TRAANG). |

### Data record for linking (TRACON) of cylinder surface transformation on main spindle (TRACYL) and inclined axis (TRAANG):

| MD24450 $MC_TRAFO_TYPE_7 | |
|---|---|
| = 8192 | Type of transformation 7 in the TRACON channel. |

| MD24454 $MC_TRAFO_GEOAX_ASSIGN_TAB_7[ ] | Assignment of geometry axes to channel axes for transformation 7. |
|---|---|
| [0] = 1 | 1. Axis = channel axis X1 |
| [1] = 3 | 2. Axis = channel axis Y1 |
| [2] = 2 | 3. Axis = channel axis Z1 |

| MD24996 $MC_TRACON_CHAIN_2[ ] | Transformation linking |
|---|---|
| [0] = 3 | Number of the TRACYL transformation (main spindle) for linking. |
| [1] = 5 | Number of the TRAANG transformation for linking. |

## 7.5.11 X zero point not at the rotation center

### General information

For some machines, the zero point of the X-axis in the Machine Coordinate System (MCS) is not located at the center of the rotating spindle.

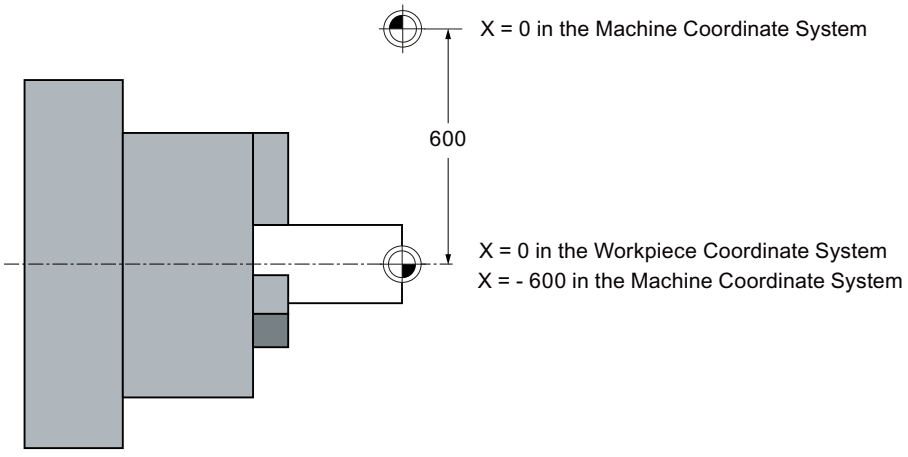



Figure 7-5 X zero point not at the rotation center

If the zero point of the X-axis in the Machine Coordinate System is not located in the center of the rotating spindle, a special commissioning is required.

You must store the position of the main spindle in the following machine data:

| MD53220 $MAS_AXIS_MCS_POSITION[0] [HSP] | Position of the axis in the MCS |
|---|---|
| = [MCS position] | e.g. - 600 |

The transformations must also be informed about the position of the rotary axis:

| MD24906 $MC_TRANSMIT_BASE_TOOL_COMP_1 | Compensation of the BASE_TOOL in frame TRANSMIT 1 |
|---|---|
| Bit 0 = 1 | MD24920 $MC_TRANSMIT_BASE_TOOL_1[0] compensated via $P_TRAFRAME . |

| MD24920 $MC_TRANSMIT_BASE_TOOL_1 [0] | Vector of the base tool for the 1st TRANSMIT transformation. |
|---|---|
| = - 600 | Value in mm |

If necessary, enter the same for further transformations (TRANSMIT, TRACYL).

Note also the following machine data:

| | |
|---|---|
| MD20360 $MC_TOOL_PARAMETER_DEF_MASK | Definition of the effect of the tool parameters. |
| Bit 3 = 0 | The work offsets in frames in the transverse axis are not included as diameter value. |

### Offset in the Z position of the counterspindle

If the zero point of the counterspindle is not located at MCS = 0, you must enter the offset in the following machine data:

| | |
|---|---|
| MD53220 $MAS_AXIS_MCS_POSITION[2] [GSP] | Position of the axis in the MCS |
| = [MCS position] | e.g 1000 |

### Fixed counterspindle

If the counterspindle does not have a traversing axis, you must enter the position of the counterspindle in the following machine data:

| | |
|---|---|
| MD53220 $MAS_AXIS_MCS_POSITION[2] [GSP] | Position of the axis in the MCS |
| = [MCS position] | e.g 800 |

### Any rotary axis

If the machine has an arbitrary rotary axis that is not further defined (e.g. by the tool carrier), you must enter the position in the following machine data:

| | |
|---|---|
| MD53220 $MAS_AXIS_MCS_POSITION[n] [RAX] | Position of the axis in the MCS<br>[n] 0 = X, 1 = Y, 2 = Z |

## 7.6 grinding

### General

Grinding tools are identified by tool type 400, 410; dressing tools are identified by tool type 490. The "New tool" function offers these types for selection. The tool list shows the "Grinding disk" and "Dressing tool" tool types listed according to their tool type. To generate the grinding disk correction, a tool setter creates a dressing program. Depending on the complexity, the tool setter makes use of external or internal code generators. This program will be assigned later to a grinding disk.

## Base variant

The dressing is performed in the JOG and AUTOMATIC operating modes, where in automatic operation, the dressing is called from a part program. To parameterize the dressing, a diagnostic screen form deployed in manual mode and in the program editor is provided.

The base package offers predefined write contours. The dressing is supported with supplied screen forms and system cycles. These system cycles are available independent of and separate from the machining cycles package.

## Oscillation cycles

These grinding cycles permit the machining of workpieces with grinding machines using two geometry axes.

The machines type does not exist because the cycles create only an oscillating movement that can be used as required on every machine.

This requires a SINUMERIK control as well as fast inputs/outputs for the program processing.

The cycles package provides the following cycles:

- CYCLE4071: Longitudinal grinding with infeed at the reversal point
- CYCLE4072: Longitudinal grinding with infeed at the reversal point and measurement control
- CYCLE4073: Longitudinal grinding with continuous infeed
- CYCLE4074: Longitudinal grinding with continuous infeed and measurement control
- CYCLE4075: Surface grinding with infeed at the reversal point
- CYCLE4077: Surface grinding, plunge grinding with measurement control
- CYCLE4078: Surface grinding with continuous infeed
- CYCLE4079: Surface grinding with intermittent infeed

## References

For further information about oscillating cycles, please refer to:

Programming Manual, Production Planning; Chapter "Programming cycles externally" > "Technological cycles"

## Grinding function screen

| MD51840 $MNS_GRIND_FUNCTION_MASK | | Grinding cycles function screen |
|---|---|---|
| Bit 0 | Reference point for measuring the dresser/wheel is a work offset | |
| = 0 | Deselection of the work offset as reference point | |
| = 1 | Selection of the work offset as reference point | |

| MD52840 $MCS_GRIND_FUNCTION_MASK | | Grinding cycles function screen |
|---|---|---|
| Bit 7 | Retraction when form-truing with G1 | |

| | |
|---|---|
| = 0 | Retraction when form-truing with rapid traverse (G0) |
| = 1 | Retraction when form-truing with feedrate (G1) |

| | |
|---|---|
| SD54840 $SNS_GRIND_FUNCTION_MASK | Grinding cycles function screen |
| SD55840 $SCS_GRIND_FUNCTION_MASK | Grinding cycles function screen |

### Form-truing (CYCLE495)

| | |
|---|---|
| SD55880 $SCS_GRIND_CONT_RELEASE_ANGLE | Angle of retraction |
| | This setting data is used to specify the angle of retraction from the contour during form-truing. |

| | |
|---|---|
| SD55881 $SCS_GRIND_CONT_RELEASE_DIST | Retraction distance |
| | This setting data is used to specify the distance retracted in both axes during form-truing. |

| | |
|---|---|
| SD55884 $SCS_GRIND_CONT_BLANK_OFFSET | Blank allowance |
| | This setting data is used to specify the distance to the grinding wheel as of which the switch-over is made from G0 to G1 during form-truing. |

# 7.7 Swiveling

## 7.7.1 Technology cycles for swiveling

### Requirement

**Only applies for technology milling.**

The commissioning of the kinematic chain of the machine is a mandatory requirement for correct swiveling functionality (CYCLE800). The kinematic chain is stored in the tool parameters $TC_CARR1 to $TC_CARR65.

**Note**

The vectors of the kinematic chain can be determined with the "Measure kinematics" measuring function (CYCLE996).

A tool carrier is available in the 828D for activating the swivel function.

| | |
|---|---|
| MD18088 $MN_MM_NUM_TOOL_CARRIER | Maximum number of definable toolholders |
| Default, cannot be changed | |

| MD28082 $MC_MM_SYSTEM_FRAME_MASK | System frames (SRAM) |
|---|---|
| Default, cannot be changed | |

## Activating the swivel function

The swivel function is enabled on the user interface via the following channel-machine data:

| MD52212 $MCS_FUNCTION_MASK_TECH | Cross-technology function screen |
|---|---|
| Bit 0 = 1 | Enable swivel |

## Configuring the input dialog

You can configure the input screen for swiveling using the following channel-specific cycle setting data. The setting data is effective for all of the declared swivel data sets.

| SD55221 $SCS_FUNKTION_MASK_SWIVEL_SET | Function screen, swiveling CYCLE800 |
|---|---|
| Bit 0 | Input field "No swiveling" |
| = 0 | Hide |
| = 1 | Display |
| Bit 1 | Selection text in retraction "Z", "ZXY" or "Fixed point 1/2" |
| = 0 | Display text Z = "Z", display text ZXY = "ZXY" |
| = 1 | Selection text retraction "Fixed point 1" or retraction "Fixed point 2".<br>If you modify the retraction variants "Z" or "ZXY" over the manufacturer cycle CUST_800.SPF, the neutral texts "Fixed point 1" and "Fixed point 2" are displayed in this case. |
| Bit 2 | Select, permit "deselection" of the swivel data set |
| = 0 | If deselection is not permitted, the "Swivel data set" (TC) selection field is not displayed in the "Swivel" input screen. |
| = 1 | Deselection permitted<br>see also swivel data set parameter $TC_CARR37<br>HUNDRED MILLIONS position |
| Bit 3 | Displays the active swivel plane under swivel in JOG. The setting of the Swivel function screen influences the swivel data set. |
| Bit 4 | Positioning in the basic position (pole position) of the kinematics |
| = 0 | Evaluation of the input values in pole position of the machine kinematics |
| = 1 | Compatibility (no evaluation of the input values when swiveling in pole position) |
| Bit 6 | Do not list swivel mode "direct" under swivel in JOG |
| Bit 7 | Swivel plane: Direction selection in basic position of the kinematic |

## Additional settings

For the swivel function, set the following machine data as a minimum to the following - although this deviates from the default value:

| MD10602 $MN_FRAME_GEOAX_CHANGE_MODE | Frames when switching over geometry axes |
|---|---|
| = 1 | The current total frame (zero point offsets) is recalculated when switching over geometry axes whereby the translations, scalings and reflections of the new geometry axes become active. The rotations of the old geometry axes are retained. |

| MD11450 $MN_SEARCH_RUN_MODE | Parameterization block search |
|---|---|
| Bit 1 = 1 | Activate PROG_EVENT.SPF after block search. This means that for a block search, the rotary axes of the active swivel data set are pre-positioned. |

| MD20110 $MC_RESET_MODE_MASK | Control basic position after RESET and end of part program |
|---|---|
| Bit 14 = 1 | System behavior "Baseframe" |

| MD20112 $MC_START_MODE_MASK | Control basic position after part program start |
|---|---|
| 400H | |

| MD20126 $MC_TOOL_CARRIER_RESET_VALUE | Active toolholder at RESET |
|---|---|
| = 0 | No active toolholder at RESET |
| = > 0 (n) | Toolholder with number n active at RESET<br>MD20126 is written to in CYCLE800<br>CYCLE800() corresponds to deselect toolholder (MD20126 = 0) |

| MD20150 $MC_GCODE_RESET_VALUES[ ] | Delete position, G group. |
|---|---|
| [41] = 1 | Delete position, G group 42 on TCOABS |
| [51] = 2 | Delete position, G group 52 on PAROTOF |
| [52] = 1 | Delete position, G group 53 on TOROTOF |
| [52] = > 1 | Delete position, G group 53 on TOROT, TOROTY or TOROTX<br>Used for machine kinematics, types "T" and "M"<br>See parameter $TC_CARR34 |

**Note**

If, after a RESET from the NC, a frame must be calculated in the tool direction, then MD20150 $MC_GCODE_RESET_VALUES[52] can be set to a value > 1.

Application: Machine kinematics with Hirth gearing

### Note on kinematics (swivel head / mixed kinematics) with Hirth tooth system:

Depending on the active plane (G17,G18,G19), the TOROT command (or TOROTX, TOROTY) is programmed in the NCU (G group 53) to calculate the compensating frame for the Hirth gearing system in CYCLE800. If the Hirth gearing causes the programmed rotation to deviate from the possible positions of the rotary axes, a $P_TOOLFRAME compensating frame is created for swivel head and mixed kinematics (see HMI active ZO/details tool reference).

If the compensating frame must be retained after RESET or end of part program, enter the following value in the channel-specific machine data:

| MD20150 $MC_GCODE_RESET_VALUES[52] | Reset behavior of G groups |
|---|---|
| = 2 | For G17 (TOROT) |
| = 3 | For G18 (TOROTY) |
| = 4 | For G19 (TOROTX) |

| MD20196 $MC_TOCARR_ROTAX_MODE | Rotary axis mode for toolholders with orientation capability |
|---|---|
| Bit 0 = 1 | Swivel data set with a rotary axis<br>used for a rotary table with C axis |
| Bit 1 = 1 | Swivel data set with two rotary axes<br>used for standard swiveling |

| MD20360 $MC_TOOL_PARAMETER_DEF_MASK | Tool parameter setting |
|---|---|
| Bit 10 = 1 | Orientation vector remains at T0 or D0 (no tool)<br>Used for machine kinematics, types "T" and "M"<br>See parameter $TC_CARR34 |
| Bit 19 = 1 | The changed orientation vector remains at T0 or D0 (no tool)<br>Bit 19 acts only with bit 10 = 1 |

| MD21186 $MC_TOCARR_ROT_OFFSET_FROM_FR | Offset of the rotary axes for a tool holder with orientation capability from the zero offset of the rotary axis. |
|---|---|
| = 0 | In CYCLE800, the WCS is recalculated when there is a value in the zero offset (ZO) of the rotary axes.<br>If values are entered in the zero offset of the rotary axes of the active swivel data set, the basic position of the swivel data set and therefore also the basic position of the WCS can change. |
| = 1 | A value in the ZO of the rotary axes acts as offset of the tool holder that has orientation capability. The Work remains unchanged. |

MD21186 may not be rewritten in a program with call CYCLE800.

If several swivel data sets per channel are agreed and machine functions are to be started when changing the swivel heads or swivel tables, an M-command can be activated in the PLC program when changing the swivel data set.

| MD22530 $MC_TOCARR_CHANGE_M_CODE | M code for swivel data set change |
|---|---|
| = 0 | No swivel data set change |
| < 0 | M code + number of the swivel data set for the swivel data set change |

## Example

The PLC can limit or invert the spindle speed or clamp or release the rotary axes, for example, by outputting the M-commands.

| | |
|---|---|
| Number of swivel data sets in channel 1 | = 2 |
| MD22530 $MC_TOCARR_CHANGE_M_CODE | = -800 |
| Programming swivel data set 1 (TCARR=1) | = M801 |
| Programming swivel data set 2 (TCARR=2) | = M802 |

## Setting the workpiece, tool and rotary table reference

Using the following machine data, you can set the workpiece, tool and rotary table reference system frames or you can influence the behavior of the system frames.

Application: System frames can be active after Reset or Power On, in order, e.g. to retract a drill from a swiveled position without causing a collision.

| MD24006 $MC_CHSFRAME_RESET_MASK | Active system frames after RESET |
|---|---|
| Bit 4 | System frame workpiece reference |
| = 0 | Not active |
| = 1 | Remains active |

| MD24007 $MC_CHSFRAME_RESET_CLEAR_MASK | Clear system frames after RESET |
|---|---|
| Bit 4 | System frame workpiece reference |
| = 0 | Do not delete |
| = 1 | Delete |

When used for measuring or swiveling in JOG, the workpiece reference must be active at RESET and not cleared (cascaded measuring).

| MD24006 $MC_CHSFRAME_RESET_MASK | Active system frames after RESET |
|---|---|
| Bit 4 = 1 | System frame for workpiece reference remains active after RESET |

| MD24007 $MC_CHSFRAME_RESET_CLEAR_MASK | Clear system frames after RESET |
|---|---|
| Bit 4 = 0 | Do not clear system frame workpiece reference after RESET |

| MD24008 $MC_CHSFRAME_POWERON_MASK | Reset system frames after power on. |
|---|---|
| Bit 2 | System frame rotary table reference (PAROT) |
| = 0 | Do not reset |
| = 1 | Reset |
| Bit 3 | System frame tool reference (TOROT,..) |
| = 0 | Do not reset |
| = 1 | Reset |
| Bit 4 | System frame workpiece reference |
| = 0 | Do not reset |
| = 1 | Reset |

| MD24080 $MC_USER_FRAME_POWERON_MASK | Settings for settable frames. |
|---|---|
| Bit 0 | |
| = 0 | Settable zero offset via power on not active. |
| =1 | Last active settable zero offset remains active after power on if MD20152 $MC_GCODE_RESET_MODE[7] = 1 |

Application: Zero offset G5xx, including all rotations, should remain active after Power On.

| MD28082 $MC_MM_SYSTEM_FRAME_MASK | Setting-up system frames (SRAM) |
|---|---|
| Bit 2 = 1 | Rotary table reference ($P_PARTFRAME) |
| Bit 3 = 1 | Tool reference ($P_TOOLFRAME) |
| Bit 4 = 1 | Workpiece reference ($P_WPFRAME) |

| MD28083 $MC_MM_SYSTEM_DATAFRAME_MASK | Setting-up data management, system frames (SRAM) |
|---|---|
| Bit 2 = 1 | Rotary table reference ($P_PARTFRAME) |
| Bit 3 = 1 | Tool reference ($P_TOOLFRAME) |
| Bit 4 = 1 | Workpiece reference ($P_WPFRAME) |

| MD28085 $MC_MM_LINK_TOA_UNIT | Assigning a TO unit to a channel (SRAM) |
|---|---|
| If several channels are set-up, then the following can be set in the machine data:<br>The number of toolholders (see MD18088) on the TO unit per channel.<br>- OR -<br>Assigning all of the set toolholders to a channel. | |

Axial machine data for the modulo rotary axes of the swivel data set

| MD30455 $MA_MISC_FUNCTION_MASK | Axis functions |
|---|---|
| Bit 0 | Modulo rotary axis programming |
| = 0 | No modulo rotary axis programming (e.g. 0 to 359.999 degrees) |
| = 1 | Modulo rotary axis programming (e.g. -180 to 180 degrees) |
| Bit 2 | Positioning, rotary axis |

| | |
|---|---|
| = 0 | As programmed |
| = 1 | Along the shortest path<br>Application: With the setting bit 2=1, the rotary axis C travels along the shortest path for G90 with DC. For further information, refer to chapter "Manufacturer cycle CUST_800.SPF (Page 151)". |

| MD32010 $MA_JOG_VELO_RAPID[AX]<br>AX = axis name | | Rapid traverse in JOG<br>Rotary and machine axes that should travel for swiveling in JOG. |
|---|---|---|
| = 10000 | Rapid traverse in JOG mode for swivelling in JOG | |

| SD42980 $SC_TOFRAME_MODE | Setting, frame definition for TOROT, PAROT |
|---|---|
| = 2000 | Swivel (default value) |

| SD42974 $SC_TOCARR_FINE_CORRECTION | Fine offset TCARR (swivel data set) |
|---|---|
| = 0 | No fine offset of the swivel data set vectors |
| = 1 | Fine offset of the swivel data set vectors<br>The parameters of the swivel data set from $TC_CARR41[n] n…number of the swivel data set and higher apply |

## Swiveling in the JOG mode

Cycle alarms 62186 and 62187 can be hidden or displayed using the following cycle machine data:

| MD55410 $MC_MILL_SWIVEL_ALARM_MASK | | Activate fault evaluation CYCLE800 |
|---|---|---|
| Bit 0 | Activates fault 61186 | |
| = 0 | Hide fault 61186 "Active zero offset G%4 and base (base reference) contains rotations" (default setting). | |
| = 1 | Display fault 61186 | |
| Bit 1 | Activate fault 61187 | |
| = 0 | Hide fault 61187 "Active base and base reference (G500) contain rotations" (default setting). | |
| = 1 | Display fault 61187 | |

## References

- Description of Functions Basic Functions; Tool Offset (W1): Inclined surface machining with 3 + 2 axes
- Function Manual Special Functions: Multiple transformations (F2)
- Programming Manual, measuring cycles: CYCLE996

## 7.7.2 CYCLE800 checklist for the identification of the machine kinematics

**Note**

Identification of the machine kinematics (kinematic chain) according to DIN 66217 or ISO 841-2001

This checklist does not claim to be complete.

- **Do the three linear axes of the machine that are active for the transformation form an orthogonal coordinate system?** Geometry axes XYZ
- **How many swivel kinematics does the machine have?**
  Combinations of two (or one) rotary axis and the three linear axes are always formed.
- **Which kinematics type is it?**
  Swivel head, swivel table or mixed kinematics of swivel head and swivel table.
- **What are the names of the rotary axes of the kinematics?**
  Manual rotary axes are permitted and do not have to be declared in the NC.
- **What is the 1st or 2nd rotary axis of a swivel data set?**
  Rule: Rotary axis 2 is based on rotary axis 1. With mixed kinematics, rotary axis 1 is always the axis for the tool orientation.
- **Is the traversing direction of the linear axes and the rotary axes correct?** Right-hand rule
  Rule: If the linear axis or the rotary axis moves the workpiece, the direction of motion of the axis and also the sign of the rotary axis vector change.
- **What is the initial setting of the kinematics?**
  This defines the tool orientation and the plane G17, G18, G19.
- **Which rotary axis rotates around which axis of the coordinate system or the machine axis (axes)?**
  This defines the rotary axis vectors of the kinematics.
  Example 1:
  Head kinematics. Rotary axis 2 rotates around axis Y → rotary axis vector V2xyz = 0,1,0
  Example 2:
  Table kinematics. Rotary axis 1 rotates around axis X → rotary axis vector V1xyz = -1,0,0

## 7.7.3 Commissioning swivel data (kinematics chain)

### Defining a swivel data set

You must create a swivel data set for every swivel head, swivel table or each combination of both.

A swivel data set consists of the following parameters: $TC_CARR1[n] to $TC_CARR65[n]

n = number of swivel data set

The parameters of the swivel data set ($TC_CARR1[n] to $TC_CARR65[n]) can be read-in and read-out in the startup operating area. Programming with appropriate value assignment

is also possible in an NC program (manufacturer cycle). The parameters of the swivel data set are immediately effective after the program has started.

## Offset vectors I1 to I4

The vectors always contain three components which represent the reference to the machine axes (X, Y, Z). The positions in the kinematic chain are measured by the machine manufacturer; they are always relevant with respect to a swivel head / swivel table (swivel data set). Offset vectors I1 to I4 refer to the non-swiveled state of the rotary axes (machine kinematics basic setting).

The machine kinematics used do not need to be fully implemented. However, be aware that the traversing range in the swivel planes may be restricted. If machine kinematics are to be implemented with just one rotary axis, this must always be declared as the 1st rotary axis.

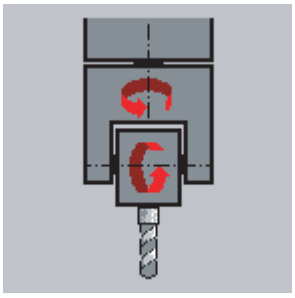
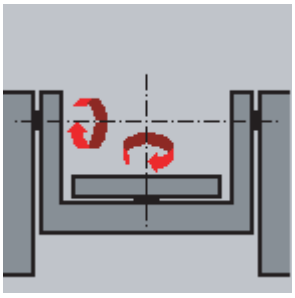
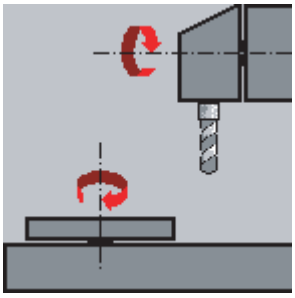
| | | | |
|---|---|---|---|
| $TC_CARR1[n], | $TC_CARR2[n], | $TC_CARR3[n] | Offset vector I1xyz |
| $TC_CARR4[n], | $TC_CARR5[n], | $TC_CARR6[n] | Offset vector I2xyz |
| | | | |
| $TC_CARR15[n], | $TC_CARR16[n], | $TC_CARR17[n] | Offset vector I3xyz |
| $TC_CARR18[n], | $TC_CARR19[n], | $TC_CARR20[n] | Offset vector I4xyz |

## Rotary axis vectors V1 and V2

| | | | |
|---|---|---|---|
| $TC_CARR7[n], | $TC_CARR8[n], | $TC_CARR9[n] | Rotary axis vector V1xyz |
| $TC_CARR10[n], | $TC_CARR11[n], | $TC_CARR12[n] | Rotary axis vector V2xyz |

## Kinematic types $TC_CARR23[n]

Following kinematics types are available for selection:

| Swivel head (type T) | Swivel table (type P) | Swivel head + swivel table (type M) |
|---|---|---|
|  |  |  |
| Offset vector I1 | Offset vector I2 | Offset vector I1 |
| Rotary axis vector V1 | Rotary axis vector V1 | Rotary axis vector V1 |
| Offset vector I2 | Offset vector I3 | Offset vector I2 |
| Rotary axis vector V2 | Rotary axis vector V2 | Offset vector I3 |
| Offset vector I3 | Offset vector I4 | Rotary axis vector V2 |
| | | Offset vector I4 |

## Name of the swivel data set

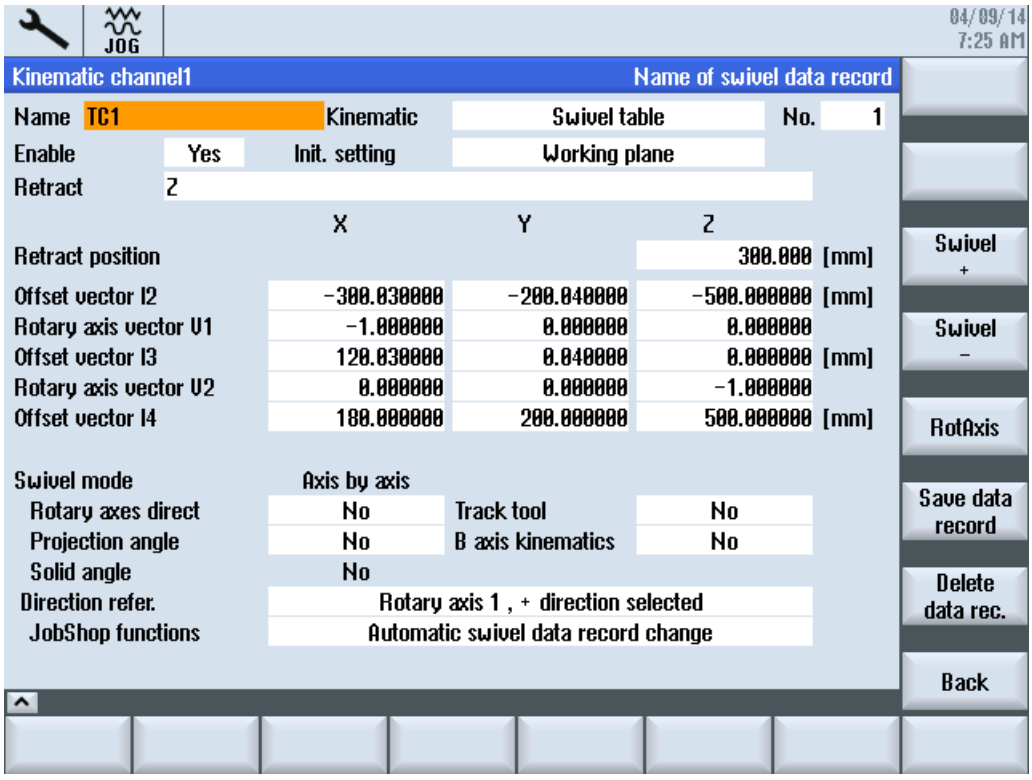



Figure 7-6 Dialog to input parameters for the swivel data set

A name is assigned to every swivel data set: $TC_CARR34[n]

The name of the swivel data set may only contain characters that are permissible for NC programming: A...Z, 0...9 and _ !

### Offset vectors I1 to I4 $TC_CARR1[n] ... $TC_CARR20[n]

The vectors always contain three components which represent the reference to the machine axes (X, Y, Z). The positions in the kinematic chain are measured by the machine manufacturer; they are always relevant with respect to a swivel head/swivel table (swivel data record).

Offset vectors I1 to I4 refer to the non-swiveled state of the rotary axes (basic setting: machine kinematics). The machine kinematics used do not need to be fully implemented. However, be aware that the traversing range in the swivel planes may be restricted. If machine kinematics are to be implemented with just one rotary axis, this must always be declared as the 1st rotary axis.

The signs of the offset vectors and the rotary axis vectors are given by the definitions of the axis directions according to **ISO 841-2001** or **DIN 66217** (right hand rule). In the case of machine kinematics that move the workpiece (rotary table), the axis direction is reversed.

The offset vectors do not have to point to the pivot point of the rotary axes. The important thing is that they point to a point on the direction of rotation (intersection).

Manually adjustable rotary axes (mode: manual) with and without measuring system are possible and are used in "simple machines".

- **Swivel head**

| | |
|---|---|
| I3 | Distance from the toolholder to the point of rotation/intersection of the 2nd rotary axis |
| I2 | Distance from the point of rotation/intersection of the 2nd rotary axis to the point of rotation/ intersection of the 1st rotary axis |
| I1 | Closing of the vector chain: I1=-(I2+I3), if swivel head cannot be changed |

- **Swivel table**

| | |
|---|---|
| I2 | Distance from the machine reference point to the point of rotation/intersection of the 1st rotary axis |
| I3 | Distance from the point of rotation/intersection of the 1st rotary axis to the point of rotation/ intersection of the 2nd rotary axis or to the reference point of the toolholder |
| I4 | Closing of the vector chain: I4=-(I2+I3), if the swivel head cannot be changed |

- **Swivel head / swivel table (mixed kinematics)**

| | |
|---|---|
| I2 | Distance from the toolholder to the point of rotation/intersection of the 1st rotary axis |
| I1 | Closing of the vector chain: I1=-I2, if the swivel head cannot be changed. |
| I3 | Distance from the machine reference point to the point of rotation/intersection of the 2nd rotary axis or to the reference point of the toolholder |
| I4 | Closing of the vector chain: I4=-I3, if the swivel head cannot be changed. |

**Fine offset of the offset vectors:**

The fine offsets act in addition to the corresponding base vectors when the Swivel function CYCLE800 or the NC function TCARR=n is called.

$TC_CARR41[n] to $TC_CARR60[n]

Assignment of the basic vectors to the vectors of the fine offset:

| | | | |
|---|---|---|---|
| **I1** | $TC_CARR1...3[n] | to | $TC_CARR41...43[n] |
| **I2** | $TC_CARR4...6[n] | to | $TC_CARR44...46[n] |
| **I3** | $TC_CARR15...17[n] | to | $TC_CARR55...57[n] |
| **I4** | $TC_CARR18...20[n] | to | $TC_CARR58...60[n] |

The fine offsets are activated by the following setting data:

| SD42974 $SC_TOCARR_FINE_CORRECTION = 1 | Fine offset TCARR on/off |
|---|---|
| = 0 | When activating a tool carrier that can be oriented, the fine offset values are not taken into account. |
| = 1 | When activating a tool carrier that can be orientated, the fine offset values are taken into account. |

## Name of rotary axes

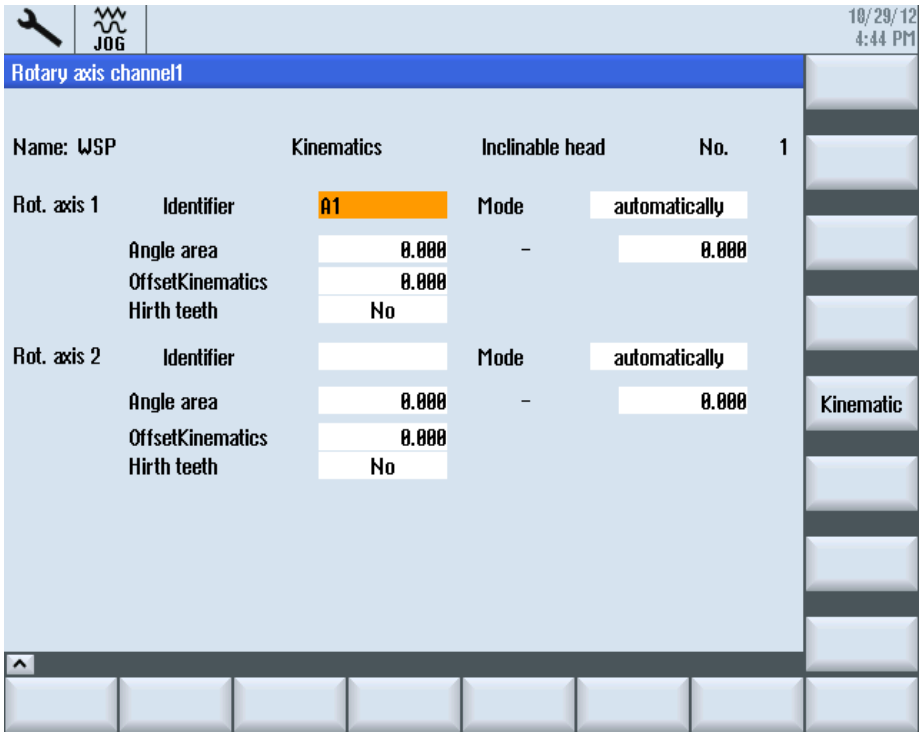



Figure 7-7 Dialog to input parameters for the rotary axis

The following identifiers should be preferably chosen for the names of the rotary axes:

- Rotary axis rotates around machine axis X → A
- Rotary axis rotates around machine axis X → B
- Rotary axis rotates around machine axis Z → C

The channel names of the corresponding NC rotary axes must be entered for automatic rotary axes: $TC_CARR37[n] in the TENS and HUNDREDS position: Mode automatic. Any axis identifiers (maximum 6 letters or numbers) can be used for manual (manually adjustable) and semiautomatic rotary axes.

$TC_CARR35[n] Name of rotary axis 1
$TC_CARR36[n] Name of rotary axis 2

### System variable $TC_CARR37[n]

The following softkeys are assigned to the Swivel function in the "Program" → "Various" operating section:

Technology milling -> "Swivel Plane" -> "Approach milling tool"

### Meaning of the decimal positions

In order that a value can be displayed in the input/ election boxes of the dialog for swiveling, the following display versions can be set:

| **$TC_CARR37[n]** | **Display variants of the input screens for CYCLE800** | |
|---|---|---|
| **Decimal place** | **Meaning** | |
| ONES | Selects the swivel mode | |
| | 0 = | Axis-by-axis |
| | 1 = | Axis-by-axis + projection angle |
| | 2 = | Axis-by-axis + projection angle + solid angle |
| | 3 = | Axis-by-axis + direct |
| | 4 = | Axis-by-axis + projection angle + direct |
| | 5 = | Axis-by-axis + projection angle + solid angle + direct |
| TENS | Rotary axis 1 | |
| | 0 = | Automatic |
| | 1 = | Manual |
| | 2 = | Semi-automatic |
| HUNDREDS | Rotary axis 2 | |
| | 0 = | Automatic |
| | 1 = | Manual |
| | 2 = | Semi-automatic |
| THOUSANDS | Selection field direction: Direction selection of the rotary axes | |
| | 0 = | No direction reference display for kinematics which only have one solution. Direction selection (_DIR) Minus is generated in the cycle call CYCLE800. |
| | 3 = | Direction reference, rotary axis 1, direction selection Minus in the basic setting of the kinematics. |
| | 4 = | Direction reference, rotary axis 2, direction selection Minus in the basic setting of the kinematics. |
| | 5 = | No direction reference display for kinematics which only have one solution. Direction selection (_DIR) Plus is generated in the cycle call CYCLE800. |
| | 8 = | Direction reference, rotary axis 1, direction selection Plus in the basic setting of the kinematics. |
| | 9 = | Direction reference, rotary axis 2, direction selection Plus in the basic setting of the kinematics. |
| | | The values 1, 2, 6 and 7 are not permitted. |
| TEN THOUSANDS | Selection field correction of the tool tip | |
| | 0 = | No display of the correction of the tool tip input field. |
| | 1 = | Correction of tool tip by means of TRAORI. |
| | 2 = | No correction of the tool tip. |
| | 3 = | Correction of the tool tip: This function assumes option "5-axis transformation (TRAORI)". |
| HUNDRED THOUSAND | reserved | |

| **$TC_CARR37[n]** | **Display variants of the input screens for CYCLE800** | |
|---|---|---|
| **Decimal place** | **Meaning** | |
| ONE MILLION / TEN MILLION | Selection field, retraction | |
| | 00 = | No retraction |
| | 01 = | Retraction Z |
| | 02 = | Retraction Z, XY |
| | 03 = | Retraction Z or Z, XY |
| | 04 = | Maximum retraction in tool direction |
| | ... | |
| | 08 = | Incremental retraction in tool direction |
| | ... | |
| | 15 = | Retraction Z or Z, XY or<br>in maximum tool direction or in incremental tool direction |
| | | $TC_CARR38[n] Retraction position X |
| | | $TC_CARR39[n] Retraction position Y |
| | | $TC_CARR40[n] Retraction position Z |
| HUNDRED MILLION | Swivel data set enabled<br>Setting swivel data set change<br>Automatic or manual tool change is only necessary under ShopMill or ShopTurn (→ see also: CUST_800.spf, markers _M2 to _M13).<br>A swivel data set must be "enabled" in every case (value ≥ 4). | |
| | 0 = | Swivel data set not enabled |
| | 4 = | Swivel data set enabled<br>Automatic swivel data set and tool change |
| | 5 = | Swivel data set enabled<br>Automatic swivel data set change and manual tool change |
| | 6 = | Swivel data set enabled<br>Manual swivel data set change and automatic tool change |
| | 7 = | Swivel data set enabled<br>Manual swivel data set and tool change |

## Retracting the geometry axes before swiveling

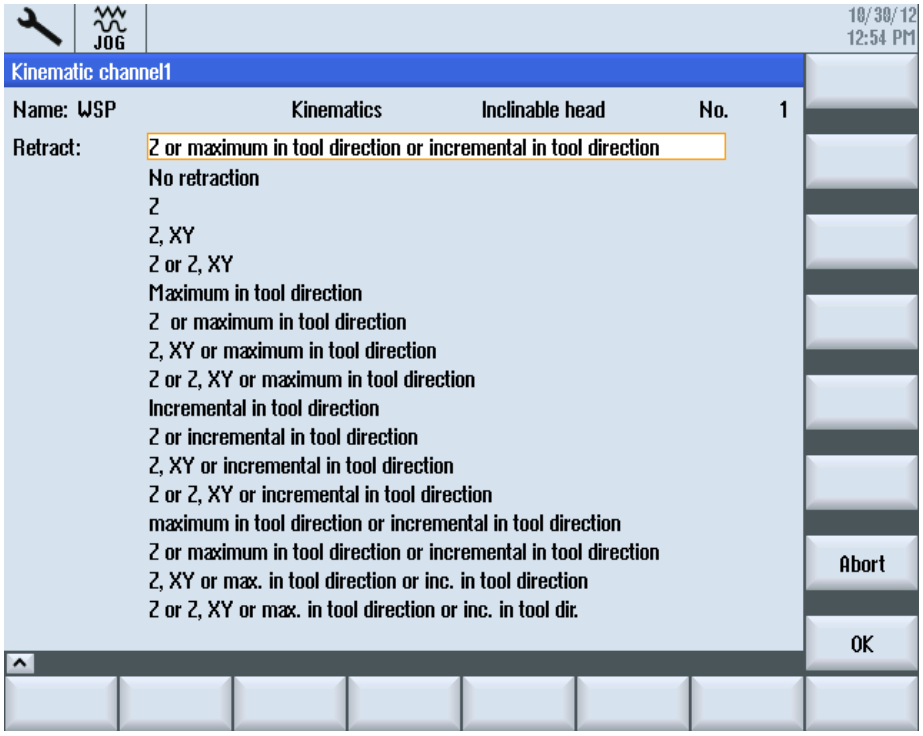



Figure 7-8 Dialog to select the type of retraction

Select the retraction variant in the input field: This is defined in the ONE MILLION and TEN MILLION position of the system variable $TC_CARR37[n]:

- Retraction of axis Z
- Retract axes Z, XY
- Retract in the tool direction, maximum or incremental

| NOTICE |
| --- |
| **Collision avoidance**<br>Make sure that the tool and the workpiece cannot collide during swivelling when moving the tool axes. |

Retracting axis Z or retracting axes Z, XY is realized as an absolute machine position at the values of parameters $TC_CARR38[n] to $TC_CARR40[n].

| | |
| --- | --- |
| $TC_CARR38[n] | Retraction position X |
| $TC_CARR39[n] | Retraction position Y |
| $TC_CARR40[n] | Retraction position Z |

The type of retraction is modified in the Manufacturer cycle CUST_800.SPF (Page 151).

## 7.7.4 Examples of machine kinematics for the commissioning of the Swivel function

### Example 1: Swivel head 1 "HEAD_1"

- Rotary axis 1(C) (manual) around Z
- Rotary axis 2(A) (manual) around X
- Manually adjustable changeable swivel head (manual)

Vectors relate to the kinematics basic setting (drawing not true-to-scale)

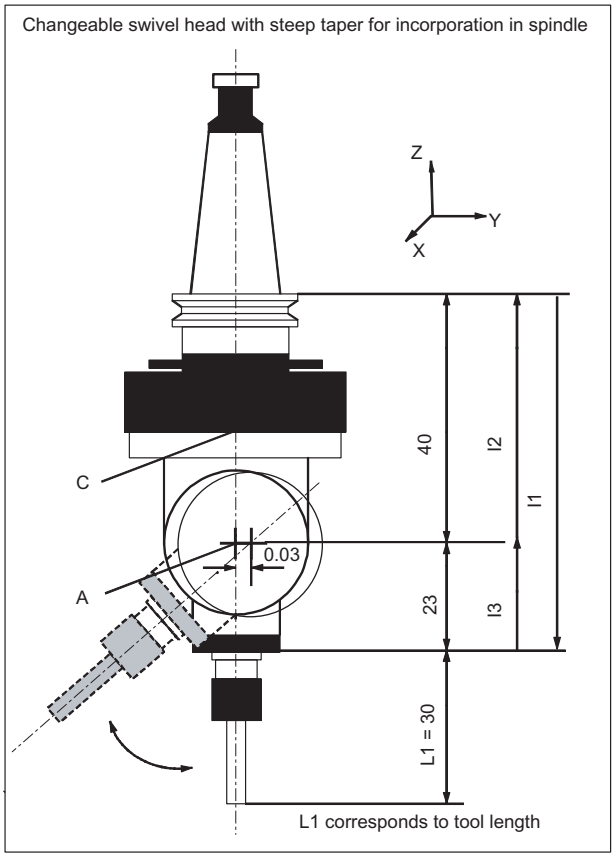



Table 7-1 Commissioning softkey "Swivel", Kinematics (Example 1)

| Kinematics | Swivel head | | HEAD_1 |
|---|---|---|---|
| Retraction | Z | | |
| | X | Y | Z |
| | | | 200.000 |
| Offset vector I1 | 0.000 | 0.030 | -63.000 |
| Rotary axis vector V1 | 0.000 | 0.000 | 1.000 |
| Offset vector I2 | 0.000 | 0.000 | 40.000 |
| Rotary axis vector V2 | 1.000 | 0.000 | 0.000 |
| Offset vector I3 | 0.000 | -0.030 | 23.000 |
| | | | |

| **Kinematics** | **Swivel head** | | **HEAD_1** |
|---|---|---|---|
| Display version | | | |
| Swivel mode | Axis-by-axis | | |
| Direction reference | Rotary axis 2 | | |
| Correct tool | No | | |
| | | | |
| **Rotary axes** | | | |
| Rotary axis 1 | C | Mode | Manual |
| Angular range | 0.000 | | 360.000 |
| Rotary axis 2 | A | Mode | Manual |
| Angular range | -15.000 | | 100.000 |

## Example 2: Swivel head 2 "HEAD_2"

| | |
|---|---|
| Rotary axis vector V1: | Rotary axis B rotates around Y |
| Rotary axis vector V2: | Rotary axis C rotates around Y **and** around Z |
| Offset vector I1: | Closure of vector chain with fixed-mounted swivel head<br>I1=-(I2+I3) |
| Offset vector I2: | Distance between pivot point of **rotary axis 1** and pivot point of **rotary axis 2** |
| Offset vector I3: | Distance between reference point of tool and pivot point of **rotary axis 2** |

Cardanic swivel head (manually adjustable) with Hirth tooth system

**Vectors relate to the basic setting of the kinematics**

If the swivel head is fixed-mounted, the vector chain is closed (see I1)

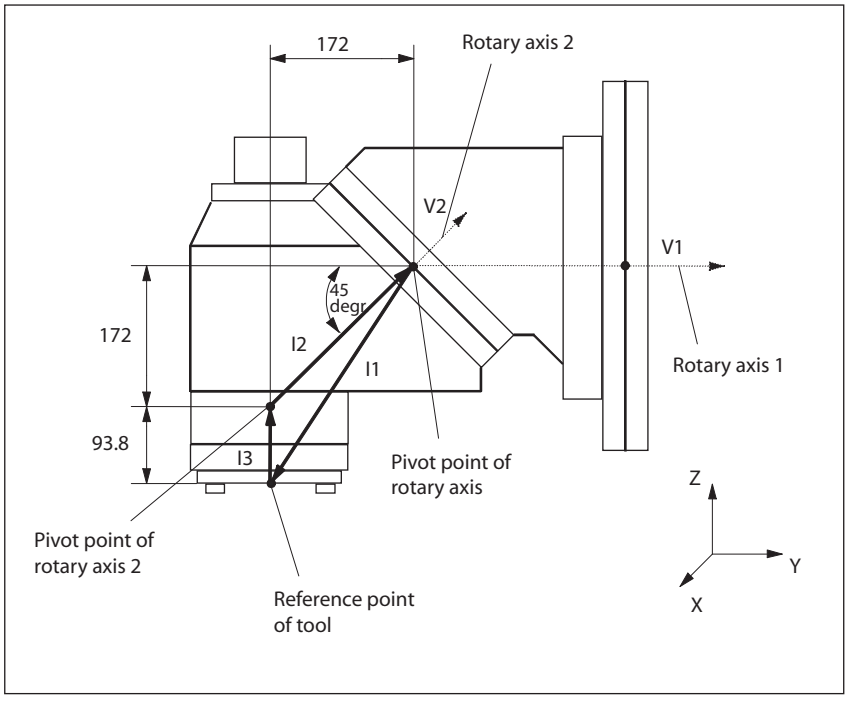



Table 7-2 Commissioning softkey "Swivel", Kinematics (Example 2)

| **Kinematics** | **Swivel head** | | **HEAD_2** |
|---|---|---|---|
| Retraction | Z | Tool direction | Max.+inc. |
| | X | Y | Z |
| | | | 200.000 |
| Offset vector I1 | 0.000 | -172.000 | -265.800 |
| Rotary axis vector V1 | 0.000 | 1.000 | 0.000 |
| Offset vector I2 | 0.000 | 172.000 | 172.000 |
| Rotary axis vector V2 | 0.000 | 1.000[1)] | 1.000[1)] |
| Offset vector I3 | 0.000 | 0.000 | 93.800 |
| | | | |
| Display version | | | |
| Swivel mode | Axis-by-axis | | |
| Direction reference | Rotary axis 2 | | |
| | | | |
| **Rotary axes** | | | |
| Rotary axis 1 | B | Mode | Manual |
| Angular range | 0.000 | | 360.000 |
| Kinematics offset | 0.000 | | |
| Hirth gearing | Yes | Angular grid | 1.000 |
| Rotary axis 2 | C | Mode | Manual |
| Angular range | 0.000 | | 180.000 |
| Kinematics offset | 0.000 | | |

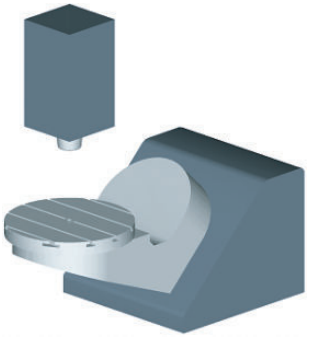
The reference point of the pivot point of rotary axes 1 and 2 can be offset on the line of rotation and does not have to coincide with the mechanical pivot point.

1) Calculation of rotary axis vector V2: 45 degree angle
   V2Y= sin(45)= 0.7071
   V2z= cos(45)= 0.7071
   V2Y and V2z can be normalized to 1.

## Example 3: Cardanic table "TABLE_45"

Vectors relate to the basic setting of the kinematics

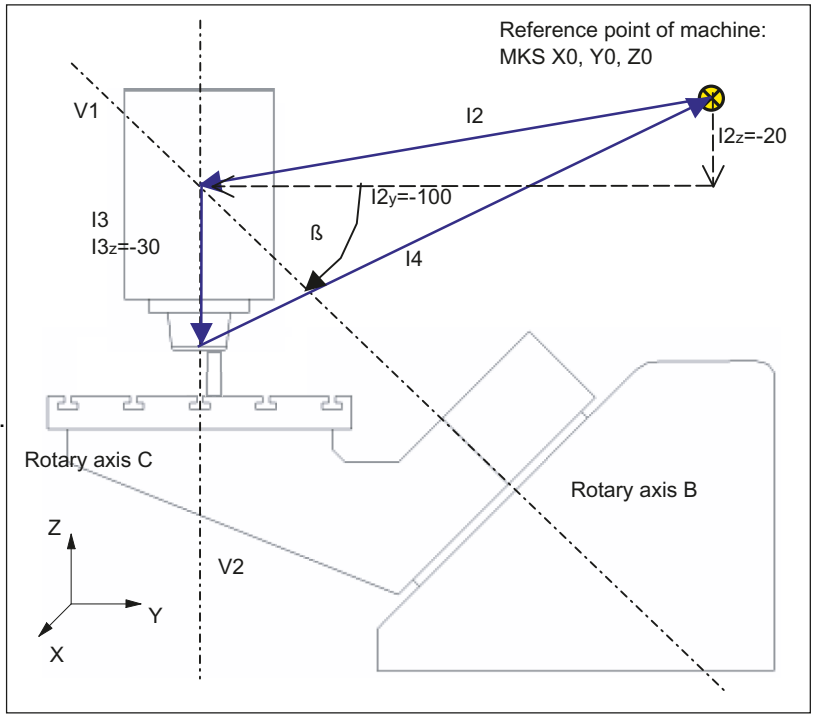
| | |
|---|---|
| Rotary axis vector V1: | Rotary axis B rotates around Y **and** around Z. |
| Rotary axis vector V2 | Rotary axis C rotates around Z. |
| Offset vector I2: | Distance from the reference point of the machine to the pivot point/intersection of **rotary axis 1** |
| Offset vector I3: | Distance between pivot point/intersection of **rotary axis 1** and pivot point/intersection of **rotary axis 2** |
| Offset vector I4: | Closure of vector chain I4=-(I2+I3) |



**Side view of the machine**

Spindle (tool adapter) is positioned on a block dimension above the top edge of the table (rotary axis C) or the center of the table.

A measuring rod in the spindle is used to determine the turning center of rotary axis C.



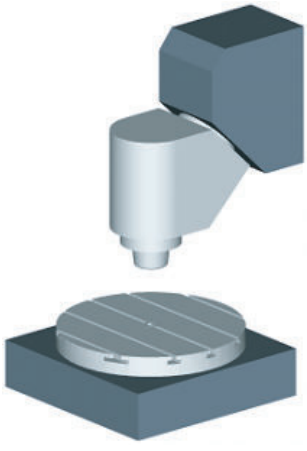
| **Kinematics** | **Swivel table** | | **TABLE_45** |
|---|---|---|---|
| | X | Y | Z |
| Offset vector I2 | 0.000 | -100.000 | -20.000 |
| Rotary axis vector V1 | 0.000 | -1.000[1)] | 1.000[1)] |
| Offset vector I3 | 0.000 | 0.000 | -30.000 |
| Rotary axis vector V2 | 0.000 | 0.000 | -1.000 |
| Offset vector I4 | 0.000 | 100.000 | 50.000 |
| | | | |
| Display version | | | |
| Swivel mode | Axis-by-axis | | |
| Direction reference | Rotary axis 2 | | |
| Correct tool | No | | |
| | | | |
| **Rotary axes** | | | |
| Rotary axis 1 | B | Mode | Auto |
| Angular range | 0.000 | | 180.000 |
| Rotary axis 2 | C | Mode | Auto |
| Angular range | 0.000 | | 360.000 |

1) Calculation of rotary axis vector V1: β = -45 degrees
V1Y= sin(-45)= -0.7071
V1z= cos(-45)= 0.7071
V1Y and V1z can be normalized to -1 and 1.

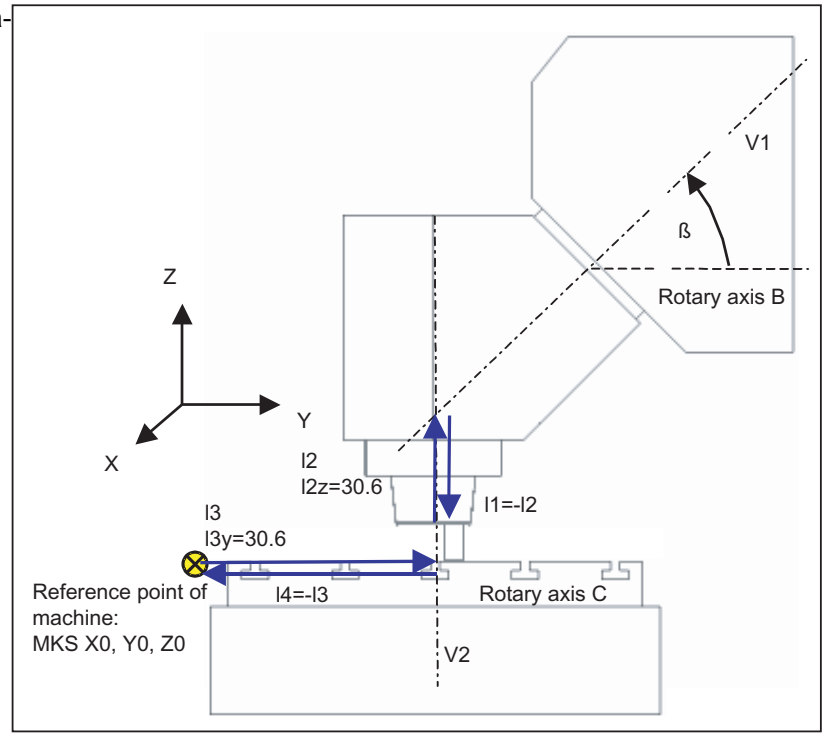
## Example 4: Swivel head/rotary table "MIXED_45"

Vectors relate to the basic setting of the kinematics

Rotary axis vector V1: Rotary axis B rotates around Y **and** around Z.
Rotary axis vector V2: Rotary axis C rotates around Z.
Offset vector I2: Distance from the reference point of the tool adapter to the pivot point/intersection of **rotary axis 1**
Offset vector I1: Closure of vector chain I1=-I2
Offset vector I3: Distance from the reference point of the machine to the pivot point/intersection of **rotary axis 2**
Offset vector I4: Closure of vector chain I4=-I3



Side view of the machine:




Spindle (tool adapter) is positioned on a block dimension above the top edge of the table (rotary axis C) or the center of the table. A measuring rod in the spindle is used to determine the turning center of rotary axis C.

Table 7-3 Commissioning softkey "Swivel", Kinematics (Example 4)

| **Kinematics** | **Mixed kinematics** | | **MIXED_45** |
|---|---|---|---|
| | X | Y | Z |
| Offset vector I1 | 0.000 | 0.000 | -30.600 |
| Rotary axis vector V1 | 0.000 | 1.000[1)] | 1.000[1)] |
| Offset vector I2 | 0.000 | 0.000 | 30.600 |
| Offset vector I3 | 300.000 | 150.000 | 0.000 |
| Rotary axis vector V2 | 0.000 | 0.000 | -1.000 |
| Offset vector I4 | -300.000 | -150.000 | 0.000 |
| | | | |
| Display version | | | |
| Swivel mode | Axis-by-axis | | |
| Direction | Rotary axis 1 | | |
| Correct tool | Yes | | |
| | | | |
| **Rotary axes** | | | |
| Rotary axis 1 | B | Mode | Auto |
| Angular range | 0.000 | | 180.000 |

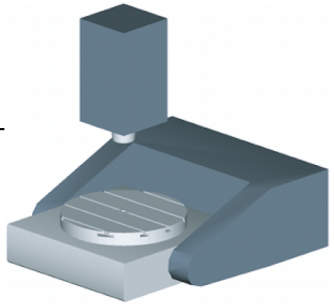
| Kinematics | Mixed kinematics | | MIXED_45 |
|---|---|---|---|
| Rotary axis 2 | C | Mode | Auto |
| Angular range | 0.000 | | 360.000 |

1) Calculation of rotary axis vector V1: β = 45 degrees
V1Y= sin(45)= -0.7071
V1z= cos(45)= 0.7071
V1Y and V1z can be normalized to 1.

## Example 5: Swivel table "TABLE_5"

Vectors relate to the basic setting of the kinematics

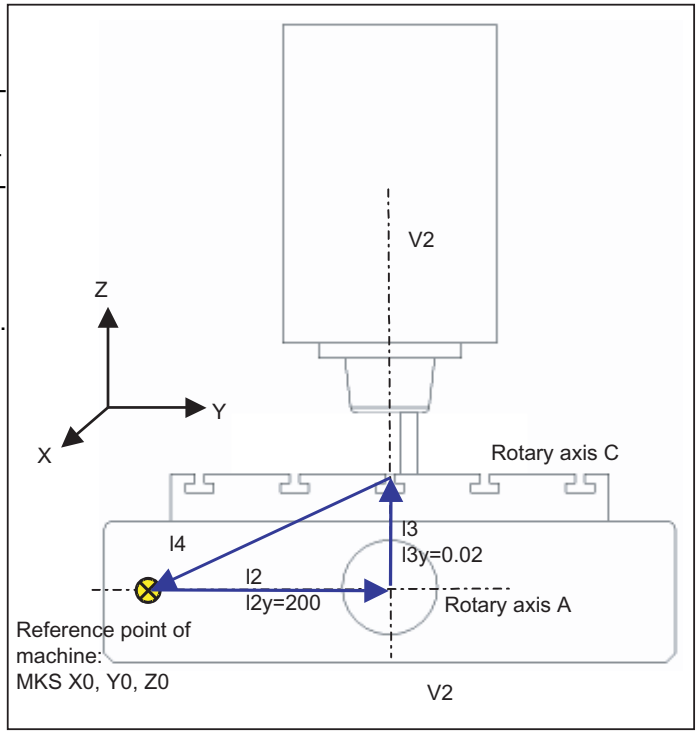
Rotary axis vector V1: Rotary axis A rotates around X.
Rotary axis vector V2: Rotary axis C rotates around Z.
Offset vector I2: Distance from the reference point of the machine to the pivot point/intersection of **rotary axis 1**
Offset vector I3: Distance from the pivot point of rotary axis 1 to the pivot point/intersection of **rotary axis 2**
Offset vector I4: Closure of vector chain I4=-(I2+I3)



**Side view of the machine from the X direction**

Spindle (tool adapter) is positioned on a block dimension above the top edge of the table (rotary axis C) or the center of the table.

A measuring rod in the spindle is used to determine the turning center of rotary axis C.



Side view of the machine from the Y direction

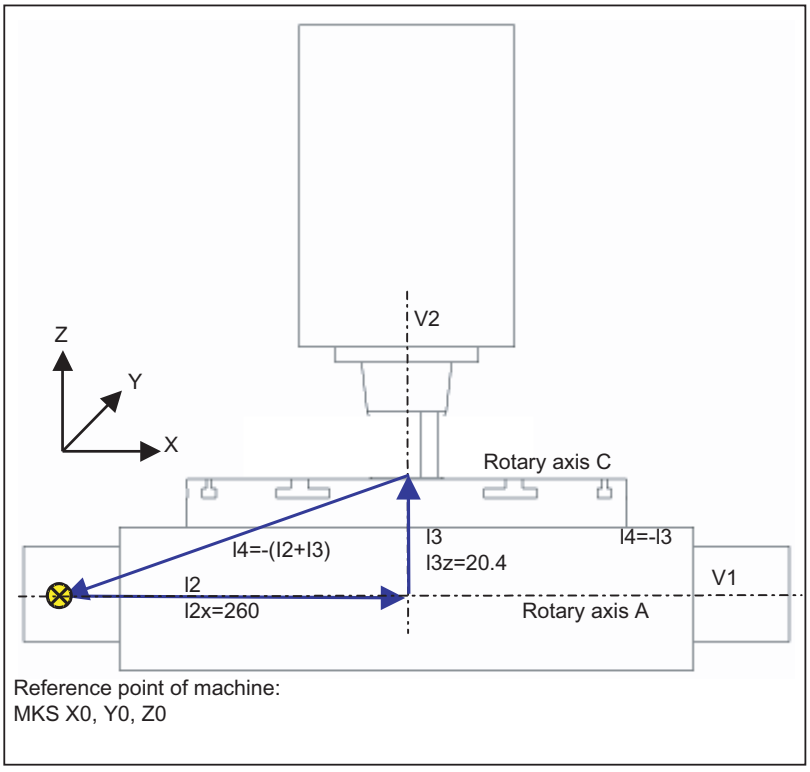



Table 7-4 Commissioning softkey "Swivel", Kinematics (Example 5)

| **Kinematics** | **Swivel table** | | **TABLE_5** |
|---|---|---|---|
| | X | Y | Z |
| Offset vector I2 | 260.000 | 200.000 | 0.000 |
| Rotary axis vector V1 | -1.000 | 0.000 | 0.000 |
| Offset vector I3 | 0.000 | 0.020 | 20.400 |
| Rotary axis vector V2 | 0.000 | 0.000 | -1.000 |
| Offset vector I4 | -260.000 | -200.020 | -20.400 |
| | | | |
| Display version | | | |
| Swivel mode | Axis-by-axis | | |
| Direction | Rotary axis 1 | | |
| Correct tool | No | | |
| | | | |
| **Rotary axes** | | | |
| Rotary axis 1 | A | Mode | Auto |
| Angular range | -90.000 | | 90.000 |
| Rotary axis 2 | C | Mode | Auto |
| Angular range | 0.000 | | 360.000 |

## 7.7.5 Manufacturer cycle CUST_800.SPF

### Function

During swiveling, all axis positions are approached using the CUST_800.SPF cycle. The call is exclusively made from the swivel cycle CYCLE800 or from the cycles E_TCARR (ShopMill) or F_TCARR (ShopTurn).

In cycle CUST_800.SPF, the function markers (_M2: to _M59) are prepared and documented. Also refer to the following "Structogram CYCLE800".

**Note**

If you modify the CUST_800, ensure that the G commands of the G groups used are restored at the end of the CUST_800 (versions S_G_1 to S_G_5!).

### Parameter

CUST_800 (INT _MODE, INT _TC1, REAL _A1, REAL _A2, INT _TC2, REAL _T_POS)

| | |
|---|---|
| _MODE | A jump is made to markers _M2 to _M59 |
| _TC1 | Number of the swivel head/table |
| _A1 | Angle of rotary axis 1 |
| _A2 | Angle of rotary axis 2 |
| _TC2 | Feed evaluation in percent (%) for swiveling in JOG mode and number of the new swivel head/table when replaced in ShopMill |
| _T_POS | Incremental position during retraction in the incremental tool direction (see marker _M44, _M45) |

### Retract prior to swiveling

If the CUST_800.SPF cycle is not modified, the Z axis (marker _M41) or the Z axis followed by the X, Y axes (marker _M42) are first traversed in the Machine to the positions when retracting prior to swiveling. The freely available position values are specified in the system variables $TC_CARR38[n] to $TC_CARR40[n]. When retracting, the active tool cutting edge is deselected (D0) and is reselected after retraction.

If retraction in the tool direction had been declared, the tool axis is retracted to the software end position (maximum in tool direction) or by an incremental distance away from the tool in the tool direction. The tool lengths are taken into account accordingly.

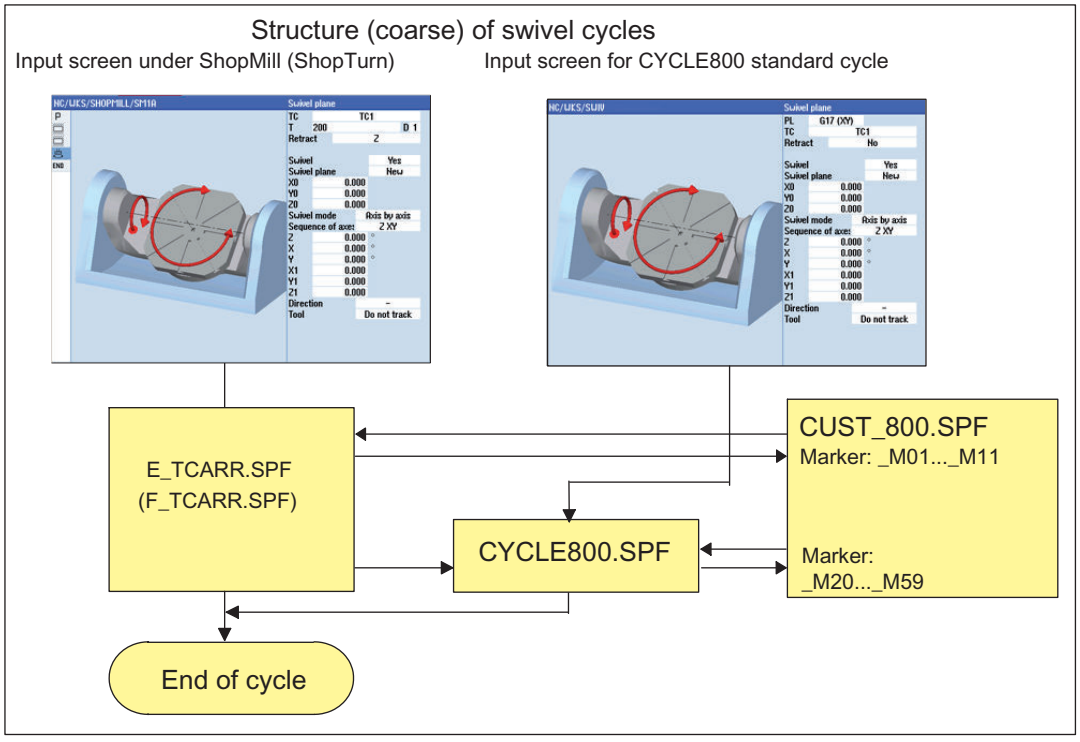



Figure 7-9 Structure (coarse) of swivel cycles

## CYCLE800 structured chart

Sequence in AUTOMATIC mode

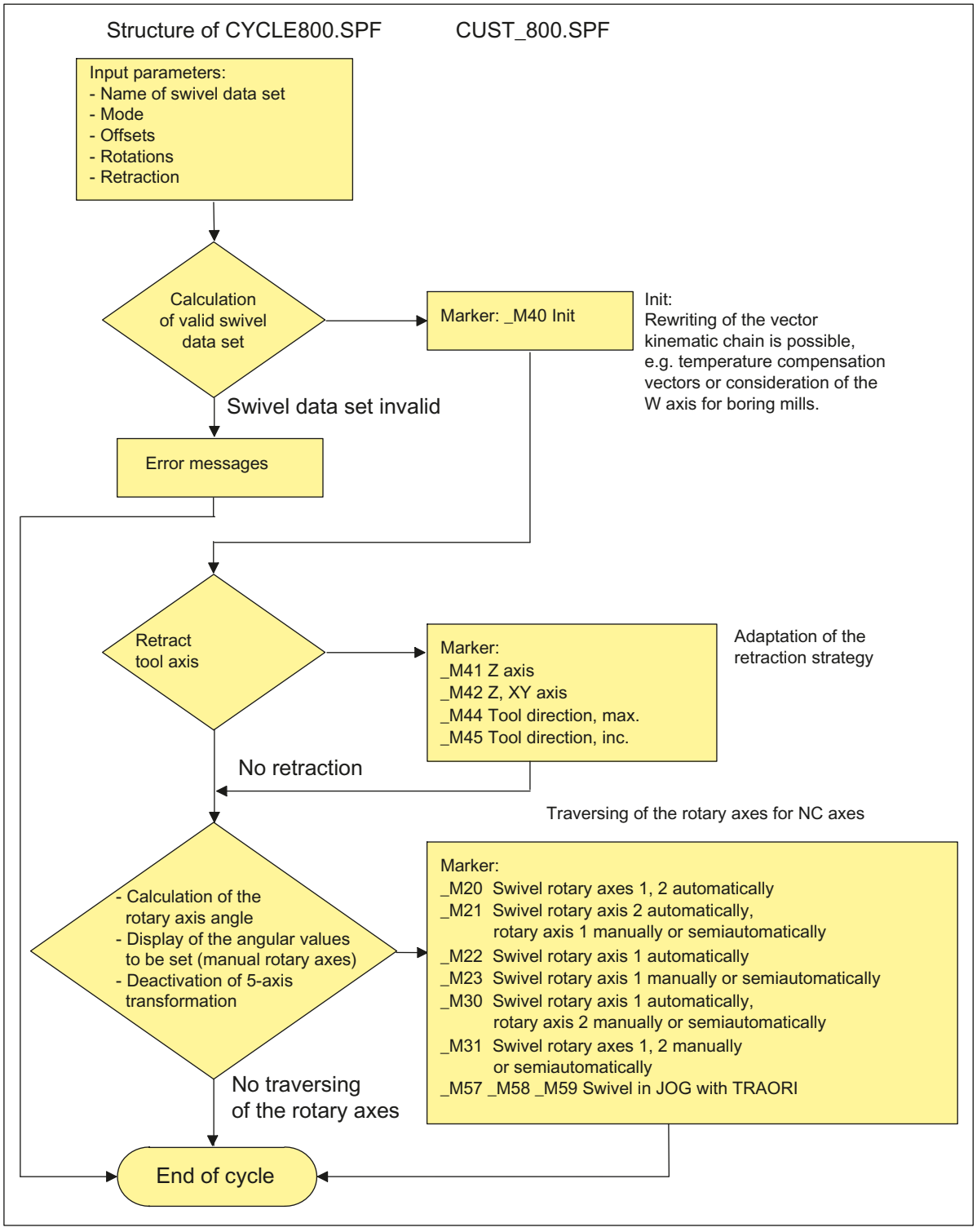



Figure 7-10 Structure: CYCLE800.SPF / CUST_800.SPF

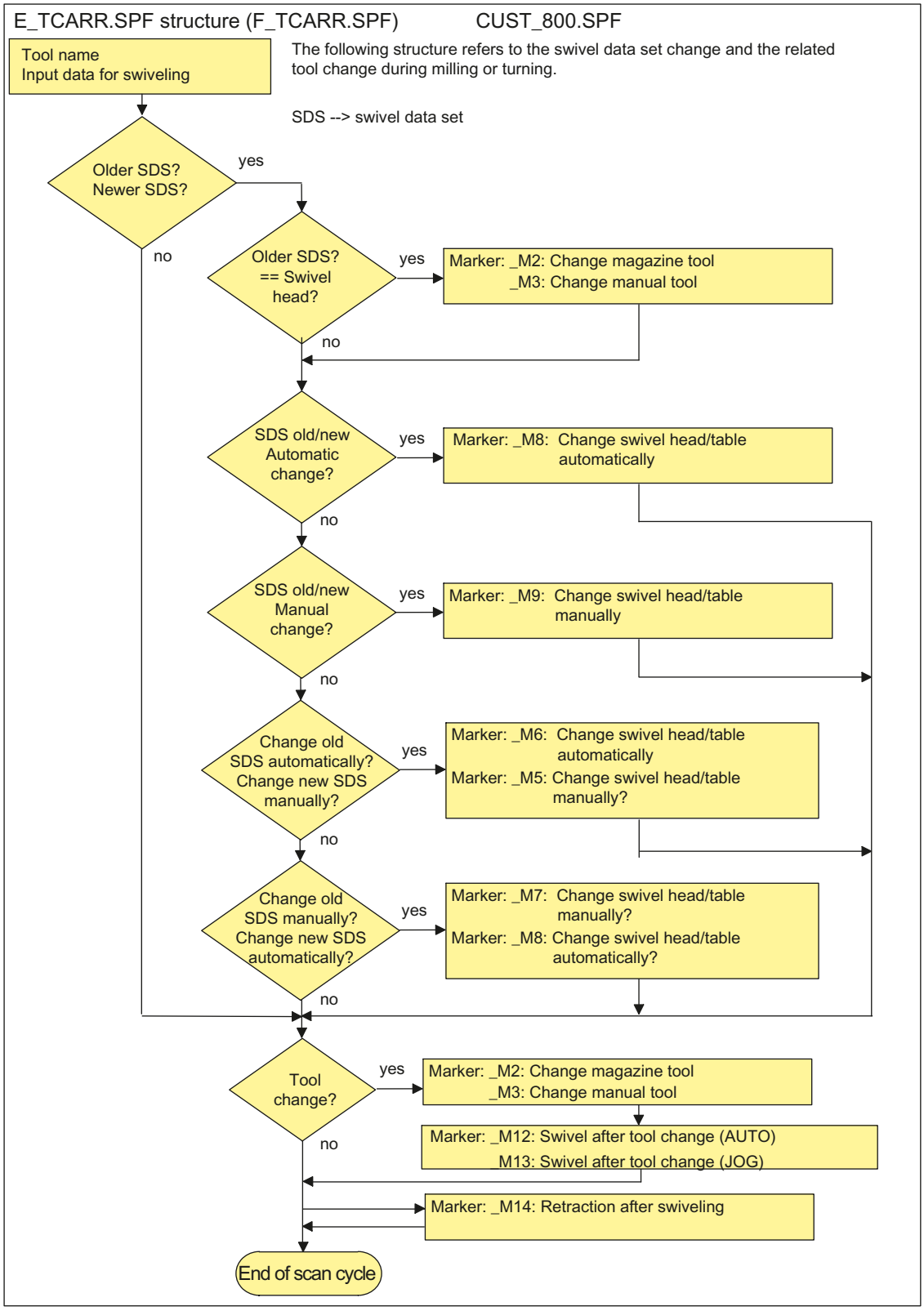



Figure 7-11 Structure: E_TCARR.SPF (F_TCARR.SPF) / CUST_800.SPF

### Note on markers _M2 to _M13

When the swivel data set or tool is changed, the linear axes are recovered using the last retraction mode (modal).

If this behavior is not desired in milling/turning, the corresponding calls must be commented out with a semicolon (;). The E_SWIV_H or F_SWIV_H cycle is called in milling/turning (see markers _M2 to _M9) in the CUST_800.SPF manufacturer cycle.

Parameter E_SWIV_H (Par1, Par2, Par3)

- Par1: Number of swivel data set (_TC1)
- Par2: Angle of 1st rotary axis
- Par3: Angle of 2nd rotary axis

### Modification examples

If the rotary axes (swivel head/table) are not to be positioned during swivel data change / tool change, the call of the E_SWIV_H cycle can be commented out at the relevant markers. If the rotary axes are to move to a certain position, an angle value can be transferred to parameters Par 2, Par 3.

### Note on markers _M14, _M15

Depending on the values of the retraction plane and the programmed swivel plane, it is possible that the linear axes now also travel the swiveled retraction plane while running up from the current position to the software limit switches after a block search. To avoid this problem, marker _M14 in the CUST_800.SPF is called after swiveling. The E_SP_RP(30) cycle preset there runs up to the milling retraction plane, whereby travel may be along the software limit switches. An appropriate retraction after block search can be set at marker _M15.

### Note on the marker _M16

If for a swivel data set change or when de-selecting the swivel data set, you wish to traverse the rotary axes to zero, the syntax of the _M16 marker can be modified accordingly, e.g. activate block N8006 E_SWIV_H(_TC1,0,0).

### Note on markers _M20 to _M31

Markers _M20 to _M31 are distinguished by machine kinematics with two rotary axes or one rotary axis. A distinction is also made between automatic rotary axes (known to the NCU) and manual (semi-automatic) rotary axes. There is only ever one valid marker for swiveling with the active swivel data set.

### Note on marker _M35

Run through _M35 for block search and a swivel data set with manual rotary axes.

### Note on marker _M40

This marker is called before the calculation and activation of the Toollcarrier function as user initialization. This makes it possible for special kinematics, to calculate or rewrite the current vectors of the active swivel data set.

Boring mill application example: For parallel axes, the vector in the Z direction can be updated with the actual value of the W axis.

A STOPRE is integrated for this marker.
Manufacturers can test as to whether this STOPRE for marker M40 in CUST_800.spf complies with their functional requirements. In this case, one must weigh up between velocity (without STOPRE) and functionality for interrupt conditions (RESET) for a swivel plane. This also depends on the type of machine kinematics.

### Note on markers _M41 to _M45

The retraction strategy prior to swiveling the rotary axes can be modified at the markers _M41 to _M45.

The retraction to the MCS positions Z or Z, XY is made considering the tool length components without tool cutter deselection.

### Note on marker _M46

Retraction before swiveling after a block search can be set at marker _M46. Variable _E_VER is 1 if it is a milling technology program.

### Note on markers _M57 to _M59

Markers _M57 to _M59 are used for swiveling in JOG mode and active 5-axis transformation (TRAORI).

### Note on "Correct tool"

"Correct tool" requires that a 5-axis transformation is set up which is equivalent to the corresponding swivel data set. The programming section for "Correct tool" is integrated in the markers _M20, _M21, _M22 and _M30. The first 5-axis transformation is called with TRAORI(1).

### Note on tool change + swivel

In general, the swivel (CYCLE800) and tool change functions for a machine are independent of each other. Thus, the swiveled work plane can be retained in a technological sequence with multiple tools, e.g. centering, drilling, tapping.

If the rotary axes of the active swivel data set are involved in the mechanical sequence of the tool change or have to be retracted, this must be taken into account in the tool change program. After the tool change, the rotary axis positions are approached as prior to the tool change. If linear axes (geometry axes) are also involved in the tool change, the rotations in the NC (swivel frame) must not be deleted. Rather, the linear axes can be positioned as machine axes using the G153 or SUPA commands.

### Note on swiveling without active tool compensation

If swiveling the rotary axes without active tool cutting edge (D0) is not possible, then you can adapt this in cycle CUST_800.SPF:

```
_M40:
IF ((NOT $P_TOOL) AND _TC1)
 LOOP
 MSG ("no tool cutting edge active")
 M0
 STOPRE
 ENDLOOP
ENDIF
GOTOF_MEND
```

## 7.7.6 CYCLE996 measure workpiece kinematics

### Function

With the Measure kinematics function, it is possible to calculate the geometric vectors used to define the 5-axis transformation (TRAORI and TCARR) by measuring the position of the ball in space.

The measurement is essentially carried out by means of workpiece probes, which scan three positions of a measuring ball on each rotary axis. The ball positions can be defined in accordance with user specifications so that they correspond to the geometric ratios on the machine. The only way of setting the ball positions is to reposition the rotary axis that is to be measured in each case.

Aside from the basic mechanics of the machine, no specific knowledge is required to use CYCLE996. No dimension drawings or machine location diagrams are necessary to carry out measuring.

### Application range

The measure kinematics function (CYCLE996) can be used to determine the data that is relevant to transformations in the case of kinematic transformations that involve rotary axes (TRAORI, TCARR).

Options:

- Redetermination of swivel data sets
  - Commissioning the machine
  - Use of swivel-mounted workholders as TCARR
- Checking swivel data sets
  - Service following collisions
  - Checking the kinematics during the machining process

Kinematics with manual axes (manually adjustable rotary tables, swivel-mounted workholders) can be measured in the same way as kinematics with NC-controlled rotary axes.

When CYCLE996 is started, a swivel data set with basic data (for kinematics type) must be parameterized. The measurement itself must be carried out without an active kinematic transformation.

If, in the setting data SD55740 $MNS_MEA_FUNCTION_MASK bit 7 is set, then depending on the setting, the kinematics can be measured based on the calculated or based on the rotary axis vectors saved in the swivel data set.

## Requirements

The following requirements must be met in order to use CYCLE996 (Measure kinematics):

- SIEMENS measuring cycles package is installed
- Workpiece probe is calibrated
- Calibration ball is mounted
- oriented tool carrier is set up (MD18088 MM_NUM_TOOL_CARRIER > 0)
- The basic geometry of the machine (X, Y, Z) is rectangular and referenced
  The right angle refers to the workpiece spindle and should be preferably checked using a test mandrel.
- Defined position of the rotary axes involved in the transformation
- Defined traverse directions in compliance with the standard of all axes involved in the transformation according to ISO 841-2001 and/or DIN 66217 (righthand rule)
- Defined size of the protocol file (MD11420 $MN_LEN_PROTOCOL_FILE = 100)

The precise procedure when measuring and programming, including examples, is described in the following manual:

## References

Programming Manual, Measuring Cycles: CYCLE996

# 7.8 High Speed Cutting (HSC)

## 7.8.1 High speed settings: Configuring CYCLE832

### Function

The High Speed Settings function (CYCLE832) is used to preset data for the machining of sculptured surfaces so that optimum machining is possible. The call of CYCLE832 contains three parameters:

- Tolerance
- Machining type (technology)
- Input of the orientation tolerance
  The 3rd parameter is significant only for machines with multi-axis orientation transformation (TRAORI). By default, it is assigned the value 1.

### Tolerance / orientation tolerance

The tolerance value for the linear axes (geometry axes) is transferred to the CNC with the NC command CTOL. CTOL = root(3) * tolerance value.

Tolerance value = 0: Tolerance value (CTOL) is loaded from the following SDs:

| | |
|---|---|
| SD55446 $SCS_MILL_TOL_VALUE_ROUGH | Tolerance value for roughing CYCLE832 |
| SD55447 $SCS_MILL_TOL_VALUE_SEMIFIN | Tolerance value for semi-finishing CYCLE832 |
| SD55448 $SCS_MILL_TOL_VALUE_FINISH | Tolerance value for finishing CYCLE832 |

Example: CYCLE832(0,_ROUGH,1) → tolerance value from SD55446 is used.

### Machining type and technology G group 59

The machining types of the technology G group 59 are permanently assigned in CYCLE832 or in CUST_832.SPF:

| Machining type | Plain text | Technology G group 59 | Field index |
|---|---|---|---|
| Deselection | _OFF | DYNNORM | 0 |
| Roughing | _ROUGH | DYNROUGH | 2 |
| Rough-finishing | _SEMIFIN | DYNSEMIFIN | 3 |
| Finishing | _FINISH | DYNFINISH | 4 |
| Roughing *) | _ORI_ROUGH | DYNROUGH | 2 |
| Semi-finishing *) | _ORI_SEMIFIN | DYNSEMIFIN | 3 |
| Finishing *) | _ORI_FINISH | DYNFINISH | 4 |

*) With entry of an orientation tolerance

Dynamic parameters can be adapted to the respective machining operation with the technology G groups. Using the commands of technology G group 59, the value of the following channel and axis-specific machine data is activated using the corresponding **array index**:

| | |
|---|---|
| MD20600 $MC_MAX_PATH_JERK[0..4] | Path dependent maximum jerk. |
| MD20602 $MC_CURV_EFFECT_ON_PATH_ACCEL[0..4] | Influence of path curvature on path dynamic response. |
| MD20603 $MC_CURV_EFFECT_ON_PATH_JERK[0..4] | Influence of path curvature on path jerk. |
| MD32300 $MA_MAX_AX_ACCEL[0..4] | Maximum axis acceleration. |
| MD32310 $MA_MAX_ACCEL_OVL_FACTOR[0..4] | Overload factor for axial velocity jumps. |
| MD32431 $MA_ MAX_AX_JERK[0..4] | Maximum axial jerk for path motion. |
| MD32432 $MA_PATH_TRANS_JERK_LIM[0..4] | Maximum axial jerk at the block transition in continuous-path mode. |

**Note**

During the optimization of the machine axes, the values (note array index) of the specified machine data must be set correctly.

### Machining type, deselection

When CYCLE832 is deselected, the G groups are programmed for the settings during the program run time; these settings are declared in machine data MD20150 $MC_GCODE_RESET_VALUES[ ]. This involves the G groups 1, 4, 10, 12, 21, 24, 30, 45, 59, 61.

The contour and orientation tolerance is deactivated (CTOL=-1, OTOL=-1).

## 7.8.2 Manufacturer cycle CUST_832.SPF

### Function

In contrast to the settings (G functions) by CYCLE832.SPF, these settings can be modified in the manufacturer cycle CUST_832.SPF.

The following markers are prepared in CUST_832.SPF:

| | |
|---|---|
| `_M_NORM:` | For deselection _OFF CYCLE832 |
| `_M_FINISH:` | For finishing _FINISH and _ORI_FINISH |
| `_M_SEMIFINISH:` | For semi-finishing _SEMIFIN and _ORI_SEMIFIN |
| `_M_ROUGH:` | For roughing _ROUGH and _ORI_ROUGH |

The following G functions are programmed in CUST_832.SPF:
`SOFT`

```
COMPCAD
G645
FIFOCTRL
UPATH
;FFWON
```

Dependent on the machining type:

```
DYNNORM
DYNFINISH
DYNSEMIFIN
DYNROUGH
```

# 7.9 Measuring cycles and measurement functions

## 7.9.1 Measuring cycles and measurement functions, general

### Requirement

Two types of electronic probes are used for measuring:

- Probe to measure the workpiece
- Probe to measure the tool

The electronic probe must be connected on the PPU or the NX module.

The electronic probe is called probe in the following document

### Setting the switching behavior

You set the electrical polarity of the connected probe using the following general machine data:

| MD13200 $MN_MEAS_PROBE_LOW_ACTIVE | | Polarity change of the probe |
|---|---|---|
| = 0 | Probe in the non-deflected state, 0 V default value<br>Probe in the deflected state, 24 V | |
| = 1 | Probe in the non-deflected state, 24 V<br>Probe in the deflected state, 0 V | |

| MD13210 $MN_MEAS_TYPE | | Measuring type for distributed drives |
|---|---|---|
| = 1 | Standard value, not adjustable | |

## Testing the probe function

You can test the switching function of the probe by manually deflecting it and checking the following PLC interface signals:

| **DB2700** | **General signals from the NC<br>NC -> PLC interface** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| DBB0 | - | - | - | -- | - | - | EMERGEN-CY OFF active | - |
| DBB1 | System inch measuring system | - | - | - | - | - | Probe actuated | |
| | | | | | | | Probe 2 | Probe 1 |

To test the switching behavior and the measured value transfer, use an NC test program with, for example, the following NC commands:

| | |
|---|---|
| MEAS | Measurement with deletion of distance-to-go |
| $AC_MEA[n] | Check the switching operation, n = measurement input number |
| $AA_MW[axis name] | Measured value of the axes in workpiece coordinates |
| $AA_MM[axis name] | Measured value of the axes in machine coordinates |

## Example of a test program

```
Program code                                  Comment
%_N_PRUEF_MESSTASTER_MPF
$PATH=/_N_MPF_DIR                             ;Testing program probe connection
N00 DEF INT MTSIGNAL                          ;Bit memory to check the switching state
N05 G17 G54 T="3D_Taster" D1                  ;Select tool geometry for probe
N10 M06                                       ;Activate tool
N15 G0 G90 X0 F150                            ;Starting position and meas. velocity
N20 MEAS=1 G1 X100                            ;Measurement at measuring input 1 in the
                                              X axis
N30 MTSIGNAL=$AC_MEA[1]                       ;Switching operation at the 1st measuring
                                              input completed, ;YES / NO
N35 IF MTSIGNAL == 0 GOTOF _FEHL1             ;evaluation of the signal
N40 R1=$AA_MM[X]                              ;Save measured value in machine coordi-
                                              nates at R1
N45 R2=$AA_MW[X]                              ;Save measured value in workpiece coordi-
                                              nates at R2
N50 M0                                        ;Check measured value in R1/R2
N55 M30
N60 _FEHL1: MSG ("Probe does not switch!")
```

| Program code | Comment |
|---|---|
| N65 M0 | |
| N70 M30 | |

## General function settings for measuring cycles

| MD51740 $MNS_MEA_FUNCTION_MASK | Measuring cycles function screen |
|---|---|
| Bit 0 | Activate calibration status monitoring – workpiece measurement for measuring in automatic |
| Bit 1 | Length reference for workpiece probe, milling technology |
| = 0 | Tool length L1, referred to the center of the probe sphere |
| = 1 | Tool length L1, referred to the sphere circumference of the probe sphere |
| Bit 2 | Support of tool carriers that can be orientated for workpiece measurement with correction in a tool |
| Bit 4 | Workpiece probe is permanently attached to the machine and cannot be positioned |
| Bit 15 | Workpiece measurement, traverse measuring block with path behavior G60 with exact stop at the end of the block |
| Bit 16 | Tool, turning technology, consideration of toolholders |

**Note**

After changing this machine data, the probe must be re-calibrated.

| MD52740 $MCS_MEA_FUNCTION_MASK | Measuring cycles function screen |
|---|---|
| Bit 0 | Measuring input, workpiece probe |
| = 0 | Workpiece probe at the CNC measuring input 1, active - default |
| = 1 | Workpiece probe at the CNC measuring input 2, active |
| Bit 1 | Workpiece measurement, functional behavior of a third geometry axis (Y axis) when turning, based on the machining plane G18! |
| Bit 2 | Calculation of the correction angle with/without active orientation transformation |
| Bit 16 | Measuring input, tool probe |
| = 0 | Tool probe at the CNC measuring input 1, active - default |
| = 1 | Workpiece probe at the CNC measuring input 2, active |

| MD54740 $SNS_MEA_FUNCTION_MASK | Measuring cycles function screen |
|---|---|
| Bit 0 | Repeat measurement, workpiece measurement after exceeding a dimensional difference (parameter _TDIF) and or confidence range (parameter _TSA) |
| Bit 1 | Repetitive measurements: workpiece measurement with alarm and cycle stop at M0 |
| Bit 2 | Workpiece measurement: M0 for tolerance alarms 62304 oversize, 62305 undersize, 62306 permissible dimensional difference exceeded |
| Bit 3 | Accept calibrated workpiece probe radius in the tool data. |
| Bit 4 | For turning-milling technology, the calibration/measuring plane is different. |
| Bit 16 | Repetitive measurement: tool measurement after a dimensional difference has been fallen below (parameter _TDIF) and or confidence range (parameter _TSA) |

| | |
|---|---|
| Bit 17 | Repetitive measurement: tool measurement with alarm and cycle stop at M0 |
| Bit 18 | Workpiece measurement: M0, after the permissible dimensional difference has been exceeded |
| Bit 19 | Tool measurement, milling tool: Spindle speed reduction at the last probing |

| MD55740 $SCS_MEA_FUNCTION_MASK | Measuring cycles function screen |
|---|---|
| Bit 0 | Collision monitoring using workpiece probe for intermediate positioning |
| Bit 1 | Coupling the spindle alignment with coordinate rotation in the active plane, for workpiece measurement with multiprobe in the automatic mode |
| Bit 2 | Direction of rotation of the spindle positioning for workpiece measurement regarding active coupling of the spindle alignment with coordinate rotation in the active plane |
| Bit 3 | For workpiece measurement, number of repeat measurements if the probe does not switch |
| Bit 4 | Measuring feedrate for workpiece measurement |
| Bit 5 | Retraction velocity from the measuring point for workpiece measurement |
| Bit 6 | Workpiece measurement: probe activation/deactivation when positioning the spindle |
| Bit 7 | Measure kinematics, scaling basis for the orientation vectors |
| Bit 8 | Measure kinematics, measurement with active TRAORI or TCARR (CYCLE 800) |
| Bit 15 | Workpiece measurement: calibration in the bore with known or unknown center point, measuring in JOG |
| Bit 16 | Collision monitoring for tool measurement |
| Bit 17 | For tool measurement, number of repeat measurements if the probe does not switch |
| Bit 18 | Measuring feedrate for tool measurement |
| Bit 19 | Retraction velocity from the measuring point for tool measurement |

## 7.9.2 Manufacturer and user cycle CUST_MEACYC.SPF

### Function

The CUST_MEACYC.SPF is part of the measuring cycle functionality. It is called in every measuring cycle before and after executing the measurement task. The CUST_MEACYC.SPF acts in the same way when measuring in the JOG mode and measuring in the AUTOMATIC mode.

You can use the CUST_MEACYC.SPF to program and execute sequences that are necessary before and/or after a measurement (e.g. activating/deactivating a probe).

**Note**

Copy the CUST_MEACYC cycle from the NC data/Cycles/Standard cycles directory to the NC data/Cycles/User cycles directory before making any changes. Changes for CUST_MEACYC only take effect in this directory.

## CUST_MEACYC structogram

Sequence in the AUTOMATIC mode, using measure bore as example, CYCLE977 with probe type 712.

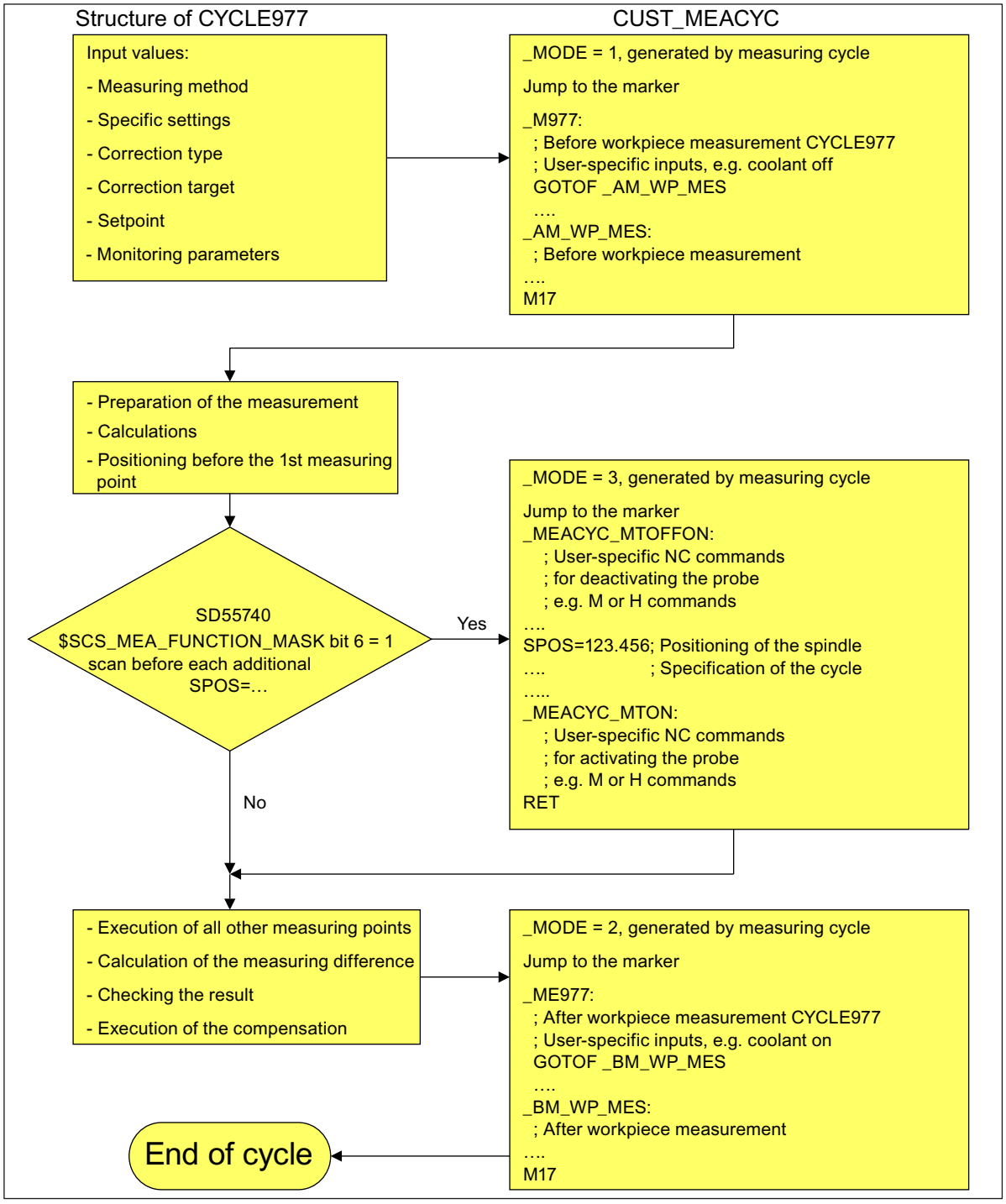



Figure 7-12 Structure: CYCLE977 / CUST_MEACYC

## 7.9.3 Measuring in the JOG mode

### Preconditions

You have already made the settings from the previous section "Measuring cycles and measurement functions, general (Page 161)".

### Workpiece measurement

**For milling:**

- The probe has been inserted in the tool spindle.
- The probe has been selected in the tool list as type 7xx (probe).
- The probe is activated as tool in the current NC channel.

### Tool measurement

To measure tools, an appropriate probe must be located in the machine space so that this can be reliably and safely reached with a tool in the spindle.

The following tool types are supported with the tool measurement:

- Milling technology: Tool types 1xx and 2xx
- Turning technology: Tool type 5xx, 1xx, 2xx

For the specified tool types, the tool lengths and the tool radii can be measured.

**Setting general data**

| SD54782 $SNS_J_MEA_FUNCTION_MASK_TOOL | | Settings for the Tool measurement in JOG input mask |
|---|---|---|
| Bit 2 | Activates the "Enable automatic tool measuring" function. | |
| Bit 3 | Selects the tool probe calibration data field, enable. | |
| Bit 10 | Selects the individual cutting edge measurement | |
| Bit 11 | Selects spindle reversal when calibrating in the plane | |

| MD51784 $MNS_J_MEA_T_PROBE_APPR_AX_DIR [i] | Measuring in JOG: Approach direction in the plane to the tool probe |
|---|---|
| -2 | Negative direction, 2nd measuring axis |
| -1 | Negative direction, 1st measuring axis (default setting) |
| 0 | Positive direction, 1st measuring axis |
| 1 | |
| 2 | Positive direction, 2nd measuring axis |

| MD52750 $MCS_J_MEA_FIXPOINT | Z value for measurement with respect to a fixed point |
|---|---|
| [0] | Position in the 1st axis |
| [1] | Position in the 2nd axis |

| MD52751 $MCS_J_MEA_MAGN_GLAS_POS | Zoom-in position for tool measurement |
|---|---|
| [0] | Position in the 1st axis |
| [1] | Position in the 2nd axis |

## Checking and setting the general machine data

| MD11450 $MN_SEARCH_RUN_MODE | Block search parameter settings |
|---|---|
| Bit 1 = 1 | Automatic ASUP start after output of action sets. Alarm 10208 is not output until the ASUP is ended (see also MD11620 $MN_PROG_EVENT_NAME). |

| MD11604 $MN_ASUP_START_PRIO_LEVEL | Priorities from which ASUP_START_MASK is effective. |
|---|---|
| = 1 - 64 | Priorities for ASUP_START_MASK. |

Setting channel-specific machine data

| MD20050 $MC_AXCONF_GEOAX_ASIGN_TAB[ ] | Assignment, geometry axis to channel axis. |
|---|---|
| [0] | For measuring in JOG mode, all of the geometry axes must be available. Preferably XYZ. |
| [1] | |
| [2] | |

| MD20110 $MC_RESET_MODE_MASK | Defines the basic control settings after reset / TP end, |
|---|---|
| = 4045H | Minimum value |
| Bit 0 = 1 | Basic control setting after power on and reset. |
| Bit 2 = 1 | |
| Bit 6 = 1 | |
| Bit 14 = 1 | |

| MD20112 $MC_START_MODE_MASK | Definition of the basic control setting after part program start. |
|---|---|
| = 400H | Minimum value |
| Bit 6 = 0 | Definition of the basic control setting after NC start. |

| MD20310 $MC_TOOL_MANAGEMENT_MASK | Activates tool manager functions |
|---|---|
| = 4002H | Minimum value |
| Bit 1 = 1 | Tool manager and monitoring functions active |
| Bit 14 = 1 | Automatic tool change for RESET and Start. |

| MD28082 $MC_SYSTEM_FRAME_MASK | System frames (SRAM). |
|---|---|
| 1 | Default, cannot be changed |

The following channel-specific machine data acts only if the system frame actual value setting and scratching is set-up using the above mentioned machine data MD28082 $MC_SYSTEM_FRAME_MASK, bit 0 = 1.

| MD24006 $MC_CHSFRAME_RESET_MASK | Active system frames after reset. |
|---|---|
| Bit 0 = 1 | Reset response of the system frames or actual value setting. |

| MD24007 $MC_CHSFRAME_RESET_CLEAR_MASK | Clear system frames on reset. |
|---|---|
| Bit 0 = 0 | Clear response of the system frames on reset. |

**Note**

If you create the conditions described in this section and you have set and checked the machine setting data, then in JOG mode, you can measure a workpiece using a workpiece probe at a milling machine!

In JOG mode, you can measure a tool using a tool probe at a milling machine or lathe!

A description is provided in the following chapters as to which settings you can make in order to adapt measuring to the specific requirements of your particular machine.

### 7.9.3.1 Measuring workpieces at the milling machines

Measuring in the "Machine" operating area can be appropriately adapted to the specific requirements using the following channel-specific general machine data and channel-specific cycle setting data.

#### General configuration machine data

| SD55630 $SCS_MEA_FEED_MEASURE | Measuring feedrate for workpiece measurement in AUTOMATIC and JOG |
|---|---|
| 300 | Default value |

**Note**

**Measuring feedrate for workpiece measuring**

All measuring cycles use the value saved in SD54611 as the measuring feedrate value following calibration of the workpiece probe. A different measuring feedrate can be assigned for each calibration field [n].

When calibrating the probe, either the measuring feedrate from SD55630 $SCS_MEA_FEED_MEASURE is used, or the measuring feedrate can be overwritten in the input screen when calibrating. To do this, SD54760 $SNS_MEA_FUNCTION_MASK_PIECE bit 4 must be set to 1.

| MD52207 $MCS_AXIS_USAGE_ATTRIB[n] | | Attributes of the axes<br>[n] channel axis number |
|---|---|---|
| Bit 6 | Request rotary axis as correction target for measuring. | |
| = 0 | Default value | |
| =1 | Rotary axis is displayed as correction target in the "Measuring in JOG" window. | |

| MD51751 $MNS_J_MEA_M_DIST_MANUELL | | Measuring path in mm, before and after the measuring point. |
|---|---|---|
| = 10 | Default value | |

| MD51757 $MNS_J_MEA_COLL_MONIT_FEED | | Position feedrate in mm/min, in the working plane for active collision monitoring. |
|---|---|---|
| = 1000 | Default value | |

| MD51758 $MNS_J_MEA_COLL_MONIT_POS_FEED | | Position feedrate in mm/min, in the infeed axis for active collision monitoring. |
|---|---|---|
| = 1000 | Default value | |

| MD51770 $MNS_J_MEA_CAL_RING_DIAM[n] | | Pre-assignment of the calibration diameter in mm specifically for the calibration data sets. |
|---|---|---|
| = -1 | Default value | |

| MD51772 $MNS_J_MEA_CAL_HEIGHT_FEEDAX[n] | | Pre-assignment of the calibration height in mm, in the infeed axis, specific for the calibration data sets. |
|---|---|---|
| = -99999 | Default value | |

### General cycle setting data

| SD54780 $SNS_J_MEA_FUNCTION_MASK_PIECE | | Configuration of the input screens for measuring in the "Machine" operating area in the JOG operating mode. |
|---|---|---|
| = 512 | Default value | |
| Bit 2 = 1 | Activates measurements with an electronic probe. | |
| Bit 3 = 1 | Selects the probe calibration data, enable. | |
| Bit 6 = 1 | Selects ZO offset in the basis reference (SETFRAME), enable. | |
| Bit 7 = 1 | Selects ZO offset in the channel-specific basic frame, enable. | |
| Bit 8 = 1 | Selects ZO offset in the global basic frame, enable. | |
| Bit 9 = 1 | Selects ZO offset in adjustable frames, enable. | |

## 7.9.3.2 Measuring tools at the milling machines

### Measuring feedrate for tool measurement in JOG and AUTOMATIC

| SD55628 $SCS_MEA_TP_FEED_MEASURE | | Calibrate measuring feedrate for tool probe and measure tool with stationary spindle. |
|---|---|---|
| = 300 | Default value | |

**Note**

**Measuring feedrate for tool measuring**

All measuring cycles use the value saved in SD54636 or SD54651 as the measuring feedrate after the tool probe has been calibrated. A different measuring feedrate can be assigned for each calibration field [n].

When calibrating the probe, either the measuring feedrate from SD55628 SCS_MEA_TP_FEED_MEASURE is used, or the measuring feedrate can be overwritten in the input screen when calibrating. To do this, SD54762 $SNS_MEA_FUNCTION_MASK_TOOL bit 4 must be set to 1.

In the following setting data, index [k] stands for the number of the current data field (probe number -1) of the probe.

| SD54633 $SNS_MEA_TP_TYPE[k] | Probe type, cube/disk |
|---|---|
| = 0 | Cube (default value) |
| = 101 | Disk in XY, working plane G17 |
| = 201 | Disk in ZX, working plane G18 |
| = 301 | Disk in YZ, working plane G19 |

The following setting data is used to define in which axes and directions it is possible to calibrate a tool probe.

| SD54632 $SNS_MEA_T_PROBE_ALLOW_AX_DIR[k] | | Axes and directions for "calibration" |
|---|---|---|
| = 133 | Default value | |

| **Decimal place** | | |
|---|---|---|
| ONES | 1. Axis | |
| | = 0 | axis not possible |
| | = 1 | only minus direction |
| | = 2 | only plus direction |
| | = 3 | both directions |
| TENS | 2. Axis | |
| | = 0 | axis not possible |
| | = 1 | only minus direction |
| | = 2 | only plus direction |
| | = 3 | both directions |
| HUNDREDS | 3. Axis | |
| | = 0 | axis not possible |
| | = 1 | only minus direction |
| | = 2 | only plus direction |
| | = 3 | both directions |

## Example

If the general setting data SD54632 $SNS_MEA_T_PROBE_ALLOW_AX_DIR[k] has the value 123, the tool probe is calibrated as follows in the G17 plane:

- X in both directions
- Y only in plus direction
- Z only in minus direction

| SD54631 $MNS_MEA_T_PROBE_DIAM_LENGTH[k] | | Effective diameter of the tool probe for length measurement. |
|---|---|---|
| = 0 | Default value | |

## General cycle machine/setting data for tool probe

| MD51780 $MNS_J_MEA_T_PROBE_DIAM_RAD[k] | | Effective diameter of the tool probe for radius measurement. |
|---|---|---|
| = 0 | Default value | |

| SD54634 $SNS_MEA_TP_CAL_MEASURE_DEPTH[n] | | Distance between the tool probe upper edge and the milling tool lower edge. |
|---|---|---|
| = 2 | Default value | |

| MD51752 $MNS_J_MEA_M_DIST_TOOL_LENGTH | | Measuring path to measure the tool length. |
|---|---|---|
| = 2 | Default value | |

| MD51753 $MNS_J_MEA_M_DIST_TOOL_RADIUS | | Measuring path to measure the tool radius. |
|---|---|---|
| = 1 | Default value | |

| MD51786 $MNS_J_MEA_T_PROBE_MEASURE_DIST | | Measuring path to calibrate the probe or for measurements with stationary spindle. |
|---|---|---|
| = 10 | Default value | |

## General cycle setting data when measuring with rotating spindle

| SD54670 $SNS_MEA_CM_MAX_PERI_SPEED[0] | | Maximum permissible peripheral speed of the tool to be measured. |
|---|---|---|
| = 100 | Default value | |

| SD54671 $SNS_MEA_CM_MAX_REVOLUTIONS[0] | | Maximum permissible tool speed of the tool to be measured. The speed is automatically reduced when exceeded. |
|---|---|---|
| = 1000 | Default value | |

| SD54672 $SNS_MEA_CM_MAX_FEEDRATE[0] | | Maximum permissible feedrate to probe the tool to be measured at the probe. |
|---|---|---|
| = 20 | Default value | |

| SD54673 SNS_MEA_CM_MIN_FEEDRATE[0] | | Minimum feedrate for the first probing of the tool to be measured at the probe. This avoids excessively small feedrates for large tool radii. |
|---|---|---|
| = 1 | Default value | |

| SD54674 $SNS_MEA_CM_SPIND_ROT_DIR[0] | | Spindle direction of rotation to measure tools. |
|---|---|---|
| = 4 | Spindle rotation the same as M4 (default value) | |

**Note**

If the spindle is already rotating when the measuring cycle is called, this direction of rotation remains independent of the setting of this data.

| SD54675 $SNS_MEA_CM_FEEDFACTOR_1[0] | Feedrate factor 1 |
|---|---|
| = 10 | Default value |
| = 0 | Only single probing with the feedrate calculated by the cycle. However, as a minimum, the value from SD54673[0] $SNS_MEA_CM_MIN_FEEDRATE. |
| >= 1 | First probing with feedrate. However, as a minimum with the value from SD54673[0] $SNS_MEA_CM_MIN_FEEDRATE)<br>SD54675[0] $SNS_MEA_CM_FEEDFACTOR_1 |

| SD54676 $SNS_MEA_CM_FEEDFACTOR_2[0] | Feedrate factor 2 |
|---|---|
| = 0 | Second probing with the feedrate calculated by the measuring cycle. This is only effective for SD54673[0] $SNS_MEA_CM_FEEDFACTOR_1 > 0, default value |
| >= 1 | Second probing with the feedrate from SD54673[0] $SNS_MEA_CM_MIN_FEEDRATE feedrate factor 2.<br>Third probing with the calculated feedrate. |

**Note**

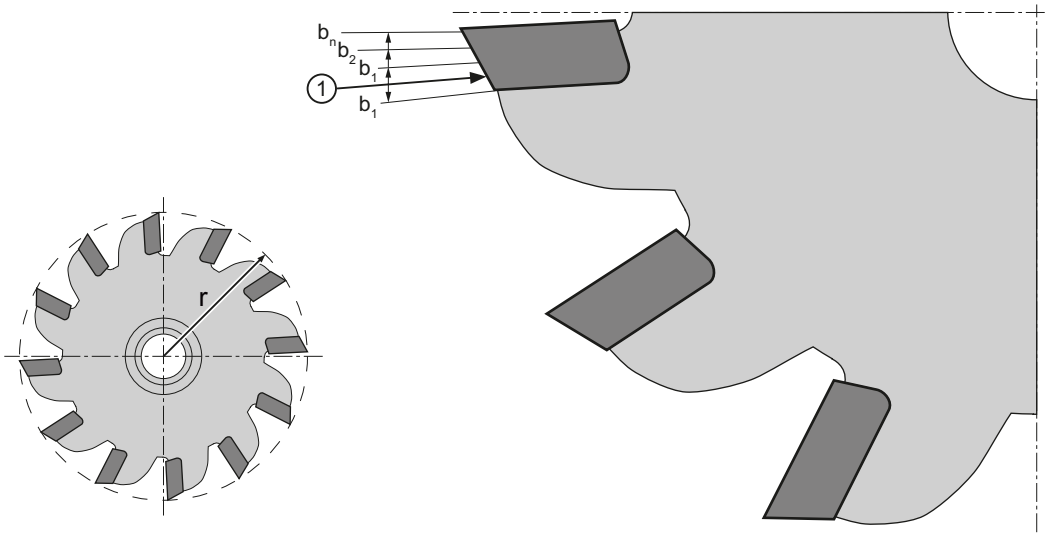
Feedrate factor 2 should be less than feedrate factor 1.

| SD54677 $SNS_MEA_CM_MEASURING_ACCURACY[0] | Specified measuring accuracy.<br>The value of this parameter always refers to the last probing of the tool at the probe! |
|---|---|
| = 0.005 | Default value |

| SD54692 $SNS_MEA_T_CIRCULAR_ARC_DIST | | Arc distance for the single cutting edge measurement of the radius. |
|---|---|---|
| = 0.25 *) | Default value | |

| SD54693 $SNS_MEA_T_MAX_STEPS | | Maximum number of probe contacts for single cutting edge measurement of the radius. |
|---|---|---|
| = 10 *) | Default value | |

*) If the value = 0, then the "highest point on the longest cutting edge" is not sought, but the measured value of the contact with the rotating spindle taken.




r Effective radius of the tool

b Arc distance = SD54692

n Maximum number of probe contacts = SD54693

① 1. Measuring point

Figure 7-13 Single cutting edge measurement

## General cycle setting data for correction using the correction tables when forming measurements with rotating spindle.

| SD54691 $SNS_MEA_T_PROBE_OFFSET | | Activate the measuring result compensation |
|---|---|---|
| = 0 | No data, default value | |
| = 1 | Correction in the cycle. This is only effective if SD54689 $SNS_MEA_T_PROBE_MANUFACTURER>0. | |
| = 2 | Correction using user-defined correction table | |

| SD54689 $SNS_MEA_T_PROBE_MANUFACTURER | | Activates pre-configured compensation tables for several tool probe models (customer-specific) |
|---|---|---|
| = 0 | No data, default value | |
| = 1 | TT130 (Heidenhain) | |
| = 2 | TS27R (Renishaw) | |

### 7.9.3.3 Measuring tools at the turning machines

#### Measuring feedrate for tool measurement in turning operation

| SD55628 $SCS_MEA_TP_FEED_MEASURE | | Calibrate measuring feedrate for tool probe and measure tool with stationary spindle. |
|---|---|---|
| = 300 | Default value | |

**Note**

**Measuring feedrate for tool measuring**

All measuring cycles use the value saved in SD54636 or SD54651 as the measuring feedrate after the tool probe has been calibrated. A different measuring feedrate can be assigned for each calibration field [n].

When calibrating the probe, either the measuring feedrate from SD55628 is used, or the measuring feedrate can be overwritten in the input screen when calibrating. To do this, SD54762 $SNS_MEA_FUNCTION_MASK_TOOL bit 4 must be set to 1.

#### General cycle machine/setting data for measuring path/feedrate

| SD42950 $SC_TOOL_LENGTH_TYP | | Assignment of the tool length offset independent of tool type. |
|---|---|---|
| = 0 | Measuring turning tools, type 5xx (default value) | |
| = 2 | Measuring turning tools, type 5xx, drilling and milling tools, type 1xx, 2xx | |

| MD51786 $MNS_J_MEA_T_PROBE_MEASURE_DIST | | Measuring path to calibrate the probe or for measurements with stationary spindle. |
|---|---|---|
| = 10 | Default value | |

| SD55628 $SCS_MEA_TP_FEED_MEASURE | | Feedrate for the calibration of a tool probe with stationary spindle. |
|---|---|---|
| = 300 | Default value | |

#### Calibrating tool probes

Use the following general cycle machine data to set the access level as of which the "Calibrate probe" softkey is displayed. This requires that the automatic tool measurement is enabled using the setting data SD54782 (bit 2 = 1).

| MD51070 $MNS_ACCESS_CAL_TOOL_PROBE | | Calibrate tool probe access level. |
|---|---|---|
| = 7 | Access rights: Access level 7 (keyswitch 0) | |

| | |
|---|---|
| = 4 | Read access rights: Access level 4 (keyswitch 3), default value |
| = 3 | Write access: Access level 3 (user) |

### 7.9.3.4 Measuring without electronic probe in JOG

#### Measure workpiece

For manual measuring functions in the JOG operating mode, for which a rotating spindle is required, the following applies:

- The channel state must be reset at the instant in time that the measured value is transferred.
- The spindle can be moved or positioned either using the T,S,M menu or using the bits in DB3800.DBB5006.
- The following preconditions apply when using the DB3800.DBB5006 technology functions with fixed speed:

| Setting the machine data | | |
|---|---|---|
| MD10709 $MN_PROG_SD_POWERON_INIT_TAB [0] = 0 | | setting data to be initialized |
| MD35035 $MA_SPIND_FUNCTION_MASK | Bit 4 = 0<br>Bit 5 = 1 | Spindle functions |
| SD43200 $SA_SPIND_S | | Speed setting value |

**Note**

The entry in SD43200 $SA_SPIND_S is kept at power on.

With these settings, the spindle velocity for manual operation, which is set in SD41200 $SN_JOG_SPIND_SET_VELO, is no longer effective.

## 7.9.4 Measuring in the AUTOMATIC mode

#### Requirement



**Software option**

In order to use the "Measuring in AUTOMATIC" function, you require the software option: "Measuring cycles"

You have already made the settings from Chapter "Measuring cycles and measurement functions, general (Page 161)".

## Workpiece measurement

**Sequence in milling technology:**

1. The probe has been selected in the tool list as type 7xx (probe).
2. The probe has been inserted in the tool spindle.
3. The probe is activated in the actual NC channel.

**Sequence in turning technology:**

1. Tool type 580 (3D probe, turning) is selected.
2. Tool is activated in the actual NC channel.

## Tool measurement

To measure tools, an appropriate probe must be located in the machine space so that this can be reliably and safely reached with a tool in the spindle.

The following tool types are supported with tool measurement:

- Milling technology: Tool types 1xx and 2xx
- Turning technology: Tool types 5xx, 1xx and 2xx

For the specified tool types, the tool lengths and the tool radii can be measured.

## Channel-specific cycle setting data

Using the following channel-specific cycle setting data, you can adapt the **workpiece and tool measuring** in the "Program" operating area to specific requirements. As a general rule, no changes are required.

| SD55613 $SCS_MEA_RESULT_DISPLAY | | Selects the screen display of the measuring result |
|---|---|---|
| = 0 | No screen display of the measuring result (default value). | |
| = 1 | Screen display of the measuring result is displayed for 8 seconds. | |
| = 3 | The measuring cycle stops an internal machine data, the measuring result is statically displayed on the screen!<br>Continue with NC start, the measuring result screen is deselected. | |
| = 4 | The measuring result is only displayed on the screen for cycle alarms 61303, 61304, 61305, 61306.<br>Continue with NC start, the measuring result display on the screen is deselected. | |

| SD55614 $SCS_MEA_RESULT_MRD | Setting of the program control of the measurement result display MRD |
|---|---|
| = 0 | Switch off the measurement result display |
| = 1 | Switch on the measurement result display |

| SD55623 $SCS_MEA_EMPIRIC_VALUE[n] | Empirical values |
|---|---|
| = 0 | Default value |

| SD55618 $SCS_MEA_SIM_ENABLE | Measuring cycle simulation |
|---|---|
| = 0 | When calling measuring cycles in conjunction with SINUMERIK Operate, ShopMill or ShopTurn simulation, the measuring cycles are skipped, default value. |
| = 1 | Default setting:<br>When calling measuring cycles in conjunction with SINUMERIK Operate, ShopMill or ShopTurn simulation, the measuring cycles are run through. However, no corrections are performed and there is no logging. There is no screen display of the measuring result. |

For further information on SD55618, please refer to:

Programming Manual, Measuring Cycles; Chapter "Behavior on block search, dry run, program testing, simulation"

### Configuring input screens for measuring cycles in the program editor

Using the following setting data, you configure the display screen and the correction options when making measurements in the AUTOMATIC mode:

- SD54760 $SNS_MEA_FUNCTION_MASK_PIECE measure workpiece: Measuring cycles in the milling area, see Chapter "Measuring workpieces, general (Page 178)"
- SD54762 $SNS_MEA_FUNCTION_MASK_TOOL measure tool: Measuring cycles in the milling area, see Chapter "Measuring tools at the milling machines (Page 182)"

**Note**

If you have created the prerequisites described in this chapter and you have set and checked the machine/setting data, you can perform measurements at the machine with a workpiece or tool probe in the AUTOMATIC mode!

If you wish to set-up workpiece measurement on a lathe, then additional settings are required. Refer to Chapter "Measuring workpieces at the turning machines (Page 188)" for the settings.

If you wish to set-up tool measurement using a tool probe, then you must also make additional settings. Please refer to Chapters "Measuring tools at the turning machines (Page 189)" or "Measuring tools at the milling machines (Page 182)" for the settings.

A description is provided in the following chapters as to which settings you can make in order to adapt measuring to the specific requirements of your particular machine.

#### 7.9.4.1 Measuring workpieces, general

Using the following general and channel-specific cycle setting data, you can appropriately adapt workpiece measurement to the specific requirements in the "Program" operating area.

## Requirement

You have already made the settings from the following section:

See also: Measuring cycles and measurement functions, general (Page 161)

## Settings

General cycle setting data

| SD54760 $SNS_MEA_FUNCTION_MASK_PIECE | | Settings for the input screen, measuring cycles in AUTOMATIC, workpiece measurement. |
|---|---|---|
| Bit 1 | Display select softkey 3D measurement | |
| Bit 3 | Enable probe calibration field to be selected | |
| Bit 4 | Select calibration input measuring feed [1)] | |
| Bit 6 | Enable ZO to be selected in the basis reference (SETFRAME) | |
| Bit 7 | Enable ZO to be selected in the channel-specific basis frame | |
| Bit 8 | Enable ZO to be selected in the basis frame | |
| Bit 9 | Enable ZO to be selected in the adjustable frame | |
| Bit 10 | Enable ZO, coarse and fine, to be selected | |
| Bit 11 | Select tool offset, geometry and wear | |
| Bit 12 | Select tool offset, not inverted and inverted | |
| Bit 13 | Select tool offset L1, R or L1, L2, L3 R | |
| Bit 14 | Select tool offset, zero offset (_TZL) | |
| Bit 15 | Select tool offset, dimensional difference monitoring (_TDIF) | |
| Bit 16 | Select workpiece measurement with spindle reversal | |
| Bit 17 | Selects align workpiece probe in the switching direction | |
| Bit 18 | Select number of measurements (_NMSP) | |
| Bit 19 | Select offset with mean value generation (_TMV) [1)] | |
| Bit 20 | Select experience values (_EVNUM) | |
| Bit 21 | Select total setting-up offset | |
| Bit 22 | Select calibration to unknown or to known center point | |
| Bit 24 | Select calibration with/without position deviation | |
| Bit 25 | Select zero offset when measuring the angulation of the spindle | |

Channel-specific cycle setting data

| SD55630 $SCS_MEA_FEED_MEASURE | | Measuring feedrate [mm/rev] |
|---|---|---|
| = 300 | Measuring feedrate when calibrating the workpiece probe | |

**Note**

**Measuring feedrate for workpiece measuring**

All measuring cycles use the value saved in SD54611 $SNS_MEA_WP_FEED[n] as measuring feedrate after the tool probe has been calibrated.

A different measuring feedrate can be assigned for each calibration field [n].

When calibrating the probe, either the measuring feedrate from SD55630 $SCS_MEA_FEED__MEASURE is used, or the measuring feedrate can be overwritten in the input screen when calibrating. To do this, SD54760 MEA_FUNCTION_MASK_PIECE bit 4 must be set to 1.

| SD55632 $SCS_MEA_FEED_RAPID_IN_PERCENT | Percentage reduction |
|---|---|
| = 50 | Default value of the percentage reduction of the rapid traverse velocity for internal cycle intermediate positioning without collision monitoring.<br>The setting of the channel-specific cycle setting data SD55600 SCS_MEA_COLLISION_MONITORING must be = 0. |

| SD55634 $SCS_MEA_FEED_PLANE_VALUE | Feedrate of the intermediate positioning in the working plane with active collision monitoring. |
|---|---|
| = 1000 | Default value |

| SD55636 $SCS_MEA_FEED_FEEDAX_VALUE | Feedrate of the intermediate positioning in the infeed axis with active collision monitoring. |
|---|---|
| = 1000 | Default value |

| SD55638 $SCS_MEA_FEED_FAST_MEASURE | Fast measuring feedrate |
|---|---|
| = 900 | Default value |

| SD55640 $SCS_MEA_FEED_CIRCLE | Feedrate with circle programming |
|---|---|
| = 1000 | Default value |

| SD55642 $SCS_MEA_EDGE_SAVE_ANG | | Safety angle when measuring a corner (CYCLE961) |
|---|---|---|
| = 0 | Degrees | |

This angle is added to the angle that the user entered between the workpiece reference edge and the 1st axis of the active coordinate system in a positive direction, and refers to the corner measurement in the automatic mode (CYCLE961).

The angle that the user enters has the designation "α0" in the parameterizing screen form, corner measurement. For compatibility programs, the designation is _STA1.

Notice:

For compatibility programs, $SCS_MEA_EDGE_SAVE_ANG should be set = 10.

### 7.9.4.2 Measuring workpieces at the milling machines

Measuring in the "Program" operating area can be adapted corresponding to the specific requirements using the channel-specific cycle setting data.

## Settings

| SD55625 $SCS_MEA_AVERAGE_VALUE[n] | | Number of mean values. |
|---|---|---|
| = 0 | Default value | |

The following cycle setting data are written to by the measuring cycles with the "Calibrate probe" measuring function. User parameterization is not necessary here. However, after the probe has been calibrated, you can check these values and if required, evaluate the probe quality, e.g. for position deviations, no values > 0.1 mm should be reached. Otherwise, the probe must be mechanically readjusted.

The measuring feedrate at the time of calibration is used for all subsequent applications of the measuring cycles.

**Note**

Observe the manufacturer's instructions for the probe.

| | |
|---|---|
| SD54600 $SNS_MEA_WP_BALL_DIAM[n] | Effective diameter of the probe ball of the workpiece probe. |
| SD54601 $SNS_MEA_WP_TRIG_MINUS_DIR_AX1[n] | Trigger point, minus direction, 1st measuring axis in the plane. |
| SD54602 $SNS_MEA_WP_TRIG_PLUS_DIR_AX1[n] | Trigger point, plus direction, 1st measuring axis in the plane. |
| SD54603 $SNS_MEA_WP_TRIG_MINUS_DIR_AX2[n] | Trigger point, minus direction, 2nd measuring axis in the plane. |
| SD54604 $SNS_MEA_WP_TRIG_PLUS_DIR_AX2[n] | Trigger point, plus direction, 2nd measuring axis in the plane. |
| SD54605 $SNS_MEA_WP_TRIG_MINUS_DIR_AX3[n] | Trigger point, minus direction, 3rd measuring axis in the tool direction. |

| | |
|---|---|
| SD54606 $SNS_MEA_WP_TRIG_PLUS_DIR_AX3[n] | Trigger point, plus direction,<br>3rd measuring axis opposite to the tool direction. In the default case = 0. |
| SD54607 $SNS_MEA_WP_POS_DEV_AX1[n] | Position deviation,<br>1st measuring axis in the plane. |
| SD54608 $SNS_MEA_WP_POS_DEV_AX2[n] | Position deviation,<br>2nd measuring axis in the plane. |
| SD54609 $SNS_MEA_WP_STATUS_RT | Calibration status of the axis positions |
| SD54610 $SNS_MEA_WP_STATUS_GEN[n] | Calibration status |
| SD54611 $SNS_MEA_WP_FEED[n] | Measuring feedrate when calibrating |

### 7.9.4.3 Measuring tools at the milling machines

#### Settings

| SD54762 $SNS_MEA_FUNCTION_MASK_TOOL | | Setting for the input screen, measuring cycles in the automatic mode, tool measurement |
|---|---|---|
| Bit 3 | Setting for the input screen, measuring cycles in the automatic mode, tool measurement, enable tool probe calibration data field to be selected | |
| Bit 4 | Select calibration input measuring feed (VMS) [1] | |
| Bit 5 | Select input feedrate and spindle speeds when probing | |
| Bit 7 | Select measurement in MCS and WCS | |
| Bit 8 | Select measurement, absolute and incremental | |
| Bit 9 | Select tool offset, geometry and wear | |
| Bit 10 | Select individual cutting edge measurement | |
| Bit 11 | Selects spindle reversal when calibrating in the plane | |
| Bit 12 | Select number of measurements (_NMSP) | |
| Bit 13 | Select experience values (_EVNUM) | |

[1] Input, measurement feedrate is valid for AUTOMATIC and JOG

#### Calibration data of the tool probe, referred to the machine coordinate system

Before calibration is started, the position of the tool probe in the machine coordinate system (MCS) must be entered into the following general cycle setting data. In this case, the reference point is the outer diameter or the tool length of the active tool in the spindle. If there is no tool in the spindle, the reference points are the spindle center point and the tool reference point at the spindle.

**Note**

**Calibrate probe**

If you have calibrated the tool probe in JOG mode, then the calibration data has already been correctly entered in: SD54632 $SNS_MEA_TP_AX_DIR_AUTO_CAL[k]

You do not need to recalibrate the tool probe in the AUTOMATIC mode.

Index [k] stands for the number of the actual data field (probe number -1).

| | |
|---|---|
| SD54625 $SNS_MEA_TP_TRIG_MINUS_DIR_AX1[k] | Trigger point of the 1st measuring axis in the negative direction. |
| SD54626 $SNS_MEA_TP_TRIG_PLUS_DIR_AX1[k] | Trigger point of the 1st measuring axis in the positive direction. |
| SD54627 $SNS_MEA_TP_TRIG_MINUS_DIR_AX2[k] | Trigger point of the 2nd measuring axis in the negative direction. |
| SD54628 $SNS_MEA_TP_TRIG_PLUS_DIR_AX2[k] | Trigger point of the 2nd measuring axis in the positive direction. |
| SD54629 $SNS_MEA_TP_TRIG_MINUS_DIR_AX3[k] | Trigger point of the 3rd measuring axis in the negative direction. |
| SD54630 $SNS_MEA_TP_TRIG_PLUS_DIR_AX3[k] | Trigger point of the 3rd measuring axis in the positive direction. |
| SD54631 $SNS_MEA_TP_EDGE_DISK_SIZE[k] | Tool probe, edge length / disk diameter. |
| SD54632 $SNS_MEA_TP_AX_DIR_AUTO_CAL[k] | Axes and directions for calibrating in AUTOMATIC mode. |
| SD54634 $SNS_MEA_TP_CAL_MEASURE_DEPTH[k] | Distance between the upper edge of the tool probe and lower edge of the tool (calibration depth, measuring depth for milling radius). |
| SD54635 $SNS_MEA_TPW_STATUS_GEN[k] | Calibration status |
| SD54636 $SNS_MEA_TPW_FEED[k] | Measuring feedrate when calibrating |

The general cycle setting data SD54632 $SNS_MEA_TP_AX_DIR_AUTO_CAL, is used to define in which axes and directions it is possible to calibrate the tool probe.

| **Decimal place** | | |
|---|---|---|
| ONES | 1st axis | |
| | = 0 | axis not possible |
| | = 1 | only minus direction |
| | = 2 | only plus direction |
| | = 3 | both directions |
| TENS | 2nd axis | |
| | = 0 | axis not possible |
| | = 1 | only minus direction |
| | = 2 | only plus direction |
| | = 3 | both directions |

| Decimal place | | |
|---|---|---|
| HUNDREDS | 3rd axis | |
| | = 0 | axis not possible |
| | = 1 | only minus direction |
| | = 2 | only plus direction |
| | = 3 | both directions |

## Example

If the general cycle machine data SD54632 $SNS_MEA_TP_AX_DIR_AUTO_CAL has the value 123, the tool probe is calibrated as follows in the G17 plane:

- X in both directions
- Y only in plus direction
- Z only in minus direction

| SD54633 $SNS_MEA_TP_TYPE[k] | Probe version |
|---|---|
| = 0 | Cube, default value. |
| = 101 | Disk in XY, working plane G17. |
| = 201 | Disk in ZX, working plane G18. |
| = 301 | Disk in YZ, working plane G19. |

## Calibration data of the tool probe referred to the workpiece coordinate system

Before calibration is started, the position of the tool probe in the workpiece coordinate system (WCS) must be roughly entered into the following general cycle setting data. In this case, the reference point is the outer diameter or the tool length of the active tool in the spindle. If there is no tool in the spindle, the reference points are the spindle center point and the tool reference point at the spindle.

### Note

When measuring tools, ensure that the data of the adjustable zero offset or the basic reference always correspond to the data when calibrating (measuring in WCS!).

Always make measurements and calibrate with the same adjustable zero offset.

| | |
|---|---|
| SD54640 $SNS_MEA_TPW_TRIG_MINUS_DIR_AX1[k] | Trigger point of the 1st measuring axis in the negative direction. |
| SD54641 $SNS_MEA_TPW_TRIG_PLUS_DIR_AX1[k] | Trigger point of the 1st measuring axis in the positive direction. |
| SD54642 $SNS_MEA_TPW_TRIG_MINUS_DIR_AX2[k] | Trigger point of the 2nd measuring axis in the negative direction. |
| SD54643 $SNS_MEA_TPW_TRIG_PLUS_DIR_AX2[k] | Trigger point of the 2nd measuring axis in the positive direction. |

| | |
|---|---|
| SD54644 $SNS_MEA_TPW_TRIG_MINUS_DIR_AX3[k] | Trigger point of the 3rd measuring axis in the negative direction. |
| SD54645 $SNS_MEA_TPW_TRIG_PLUS_DIR_AX3[k] | Trigger point of the 3rd measuring axis in the positive direction. |
| SD54646 $SNS_MEA_TPW_EDGE_DISK_SIZE[k] | Tool probe, edge length / disk diameter. |
| SD54647 $SNS_MEA_TPW_AX_DIR_AUTO_CAL[k] | Automatic calibration of tool probe, enable axes/directions |
| SD54648 $SNS_MEA_TPW_TYPE[k] | Probe version |
| = 0 | Cube (default value) |
| = 101 | Disk in XY, working plane G17. |
| = 201 | Disk in ZX, working plane G18. |
| = 301 | Disk in YZ, working plane G19. |
| SD54649 $SNS_MEA_TPW_CAL_MEASURE_DEPTH[k] | Distance between the upper edge of the tool probe and lower edge of the tool (calibration depth, measuring depth for milling radius). |
| SD54650 $SNS_MEA_TPW_STATUS_GEN[k] | Calibration status |
| SD54651 $SNS_MEA_TPW_FEED[k] | Measuring feedrate when calibrating |

The following general cycle setting data SD54647 $SNS_MEA_TPW_AX_DIR_AUTO_CAL is used to define in which axes and directions it is possible to calibrate a tool probe.

| **Decimal place** | | |
|---|---|---|
| ONES | 1. Axis | |
| | = 0 | axis not possible |
| | = 1 | only minus direction |
| | = 2 | only plus direction |
| | = 3 | both directions |
| TENS | 2. Axis | |
| | = 0 | axis not possible |
| | = 1 | only minus direction |
| | = 2 | only plus direction |
| | = 3 | both directions |
| HUNDREDS | 3. Axis | |
| | = 0 | axis not possible |
| | = 1 | only minus direction |
| | = 2 | only plus direction |
| | = 3 | both directions |

## Example

If the general cycle machine data SD54647 $SNS_MEA_TPW_AX_DIR_AUTO_CAL has the value 123, the tool probe is calibrated as follows in the G17 plane:

- X in both directions
- Y only in plus direction
- Z only in minus direction

## Monitoring when measuring with a rotating spindle

| SD54670 $SNS_MEA_CM_MAX_PERI_SPEED[0] | | Maximum permissible peripheral speed of the tool to be measured. |
|---|---|---|
| = 100 | Default value | |

| SD54671 $SNS_MEA_CM_MAX_REVOLUTIONS[0] | | Maximum permissible tool speed of the tool to be measured. The speed is automatically reduced when exceeded. |
|---|---|---|
| = 1000 | Default value | |

| SD54672 $SNS_MEA_CM_MAX_FEEDRATE[0] | | Maximum permissible feedrate to probe the tool to be measured at the probe. |
|---|---|---|
| = 20 | Default value | |

| SD54673 $SNS_MEA_CM_MIN_FEEDRATE[0] | | Minimum feedrate for the first probing of the tool to be measured at the probe. This avoids excessively small feedrates for large tool radii. |
|---|---|---|
| = 1 | Default value | |

| SD54674 $SNS_MEA_CM_SPIND_ROT_DIR[0] | | Spindle direction of rotation to measure tools. |
|---|---|---|
| 4 = M4 | Default value | |

**Note**

If the spindle is already rotating when the measuring cycle is called, this direction of rotation remains independent of the setting of this data.

| SD54675 $SNS_MEA_CM_FEEDFACTOR_1[0] | Feedrate factor 1 |
|---|---|
| = 10 | Default value |

| | |
|---|---|
| = 0 | Only single probing with the feedrate calculated by the cycle. However, as a minimum, the value from SD54673[0] $SNS_MEA_CM_MIN_FEEDRATE. |
| = ≥ 1 | First probing with feedrate. However, as a minimum with the value from SD54673[0] $SNS_MEA_CM_MIN_FEEDRATE)<br>SD54675[0] $SNS_MEA_CM_FEEDFACTOR_1 |

| SD54676 $SNS_MEA_CM_FEEDFACTOR_2[0] | Feedrate factor 2 |
|---|---|
| = 0 | Second probing with the feedrate calculated by the cycle. This is only effective for SD54673 $SNS_MEA_CM_FEEDFACTOR_1[0] > 0, default value. |
| = ≥ 1 | Second probing with the calculated feedrate from SD54673 $SNS_MEA_CM_MIN_FEEDRATE[0] feedrate factor 2.<br>Third probing with the calculated feedrate. |

**Note**

Feedrate factor 2 should be less than feedrate factor 1.

| SD54677 $SNS_MEA_CM_MEASURING_ACCURACY[0] | | Specified measuring accuracy.<br>The value of this parameter always refers to the last probing of the tool at the probe. |
|---|---|---|
| = 0.005 | Default value | |

## Measurement with rotating spindle: Measured value correction using correction tables

| SD54691 $SNS_MEA_T_PROBE_OFFSET | Activates the measuring result correction |
|---|---|
| = 0 | No data, default value |
| = 1 | Correction in the cycle. This is only effective if SD54690 $SNS_MEA_T_PROBE_MANUFACTURER>0. |
| = 2 | Correction using user-defined correction table |

| SD54689 $SNS_MEA_T_PROBE_MANUFACTURER | Activate pre-configured compensation tables for several tool probe models (customer-specific). |
|---|---|
| = 0 | No data, default value |
| = 1 | TT130 (Heidenhain) |
| = 2 | TS27R (Renishaw) |

### Correction values for users

If the general cycle setting data SD54691 $SNS_MEA_T_PROBE_OFFSET= 2, the following settings apply:

| | | |
|---|---|---|
| SD54695 to SD54700 | Correction values for radius measurement. | See the subsequent general cycle setting data. |
| SD54705 to SD54710 | Correction values for length measurement. | See the subsequent general cycle setting data. |

| SD54695 $SNS_MEA_RESULT_OFFSET_TAB_RAD1[n]<br>SD54705 $SNS_MEA_RESULT_OFFSET_TAB_LEN1[n] | Radius measurement<br>Length measurement |
|---|---|
| = 0 | 0 |
| = 1 | 1. Radius |
| = 2 | 2. Radius |
| = 3 | 3. Radius |
| = 4 | 4. Radius |

## 7.9.4.4 Measuring workpieces at the turning machines

### General settings for turning

| MD52740 $MNS_MEA_FUNCTION_MASK | Measuring cycles function screen |
|---|---|
| Bit 1 | Workpiece measurement, measuring in the third geometry axis (Y, when turning |
| = 0 | The measuring cycles do not support a third geometry axis (Y axis)! |
| =1 | Setpoint input and parameterization (SETVAL, _TUL, _TLL, SZO) refer to the third geometry axis (Y axis). The correction of the tool length or zero offset is however realized in the second geometry axis (X axis, ordinate) active components (i.e. measure in Y and correct in X). The correction target can be influenced using parameter _KNUM! |
| Bit 16 | Measuring input, tool probe |

### Calibration data of the tool probe, referred to the machine coordinate system

Before calibration is started, the position of the tool probe in the machine coordinate system (MCS) must be entered into the following general cycle setting data.

| | |
|---|---|
| SD54615 $SNS_MEA_CAL_EDGE_BASE_AX1[n] | Calibration slot base referred to the 1st measuring axis. |
| SD54617 $SNS_MEA_CAL_EDGE_PLUS_DIR_AX1[n] | Calibration slot edge in the positive direction of the 1st measuring axis. |
| SD54618 $SNS_MEA_CAL_EDGE_MINUS_DIR_AX1[n] | Calibration slot edge in the negative direction of the 1st measuring axis. |
| SD54619 $SNS_MEA_CAL_EDGE_BASE_AX2[n] | Calibration slot base referred to the 2nd measuring axis. |
| SD54620 $SNS_MEA_CAL_EDGE_UPPERE_AX2[n] | Upper calibration slot edge referred to the 2nd measuring axis. |

| | |
|---|---|
| SD54621 $SNS_MEA_CAL_EDGE_PLUS_DIR_AX2[n] | Calibration slot edge in the positive direction of the 2nd measuring axis. |
| SD54622 $SNS_MEA_CAL_EDGE_MINUS_DIR_AX2[n] | Calibration slot edge in the negative direction of the 2nd measuring axis. |

**Note**

For a standard lathe with axes X and Z (G18), axis Z is the 1st measuring axis and axis X is the 2nd measuring axis.

### 7.9.4.5 Measuring tools at the turning machines

#### Calibration data of the tool probe referred to the machine coordinate system

If you wish to calibrate the tool probe in the machine coordinate system, then the position of the tool probe in the machine coordinate system must be entered into the following general cycle setting data.

| | |
|---|---|
| SD54625 $SNS_MEA_TP_TRIG_MINUS_DIR_AX1[k] | Trigger point in minus direction of the 1st measuring axis (for G18 Z) |
| SD54626 $SNS_MEA_TP_TRIG_PLUS_DIR_AX1[k] | Trigger point in plus direction of the 1st measuring axis (for G18 Z) |
| SD54627 $SNS_MEA_TP_TRIG_MINUS_DIR_AX2[k] | Trigger point in minus direction of the 2nd measuring axis (for G18 X). |
| SD54628 $SNS_MEA_TP_TRIG_PLUS_DIR_AX2[k] | Trigger point in plus direction of the 2nd measuring axis (for G18 X). |

#### Calibration data of the tool probe referred to the workpiece coordinate system

If you wish to calibrate the tool probe in the workpiece coordinate system, then the position of the tool probe in the workpiece coordinate system must be entered into the following general cycle setting data. In this case, the reference point is the outer diameter or the tool length of the active tool in the spindle.

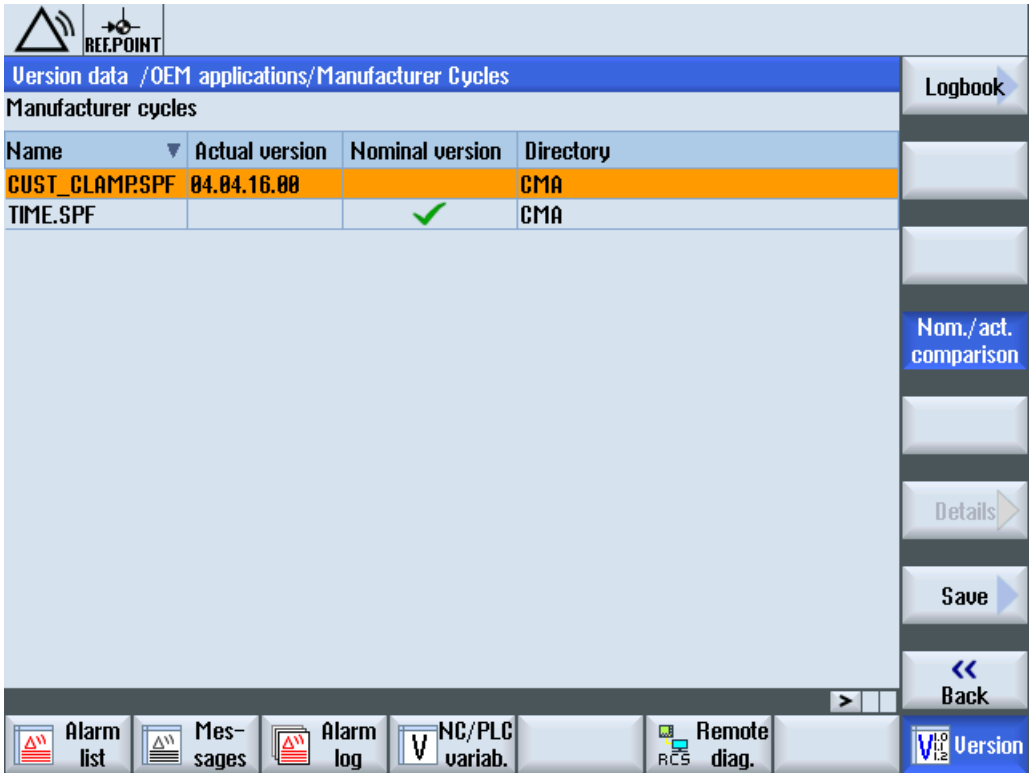
Index [k] stands for the number of the actual data field (probe number -1).

| | |
|---|---|
| SD54640 $SNS_MEA_TPW_TRIG_MINUS_DIR_AX1[k] | Trigger point minus direction of the 1st measuring axis (for G18 Z). |
| SD54641 $SNS_MEA_TPW_TRIG_PLUS_DIR_AX1[k] | Trigger point plus direction of the 1st measuring axis (for G18 Z). |
| SD54642 $SNS_MEA_TPW_TRIG_MINUS_DIR_AX2[k] | Trigger point minus direction of the 2nd measuring axis (for G18 X). |
| SD54643 $SNS_MEA_TPW_TRIG_PLUS_DIR_AX2[k] | Trigger point plus direction of the 2nd measuring axis (for G18 X). |

# 7.10 Compare cycles version

## 7.10.1 Display cycles version

If cycles have version details, you can display them in the version screen.




### Precondition

The version details are included in the cycles files in the following form:

`;VERSION:` <Version> `;DATE:` <YYYY-MM-DD>

**Example:**

```
;VERSION: 05.05.05.00 ;DATE: 2012-11-30
```

### Procedure




1. Select the "Diagnostics" operating area.




2. Press the "Version" softkey.

   It takes some time to call the version display. While the version data is being determined a progress message box and the appropriate text are displayed in the dialog line.




3. Select the "OEM applications /manufacturer cycles" areas and press the "Details" softkey.

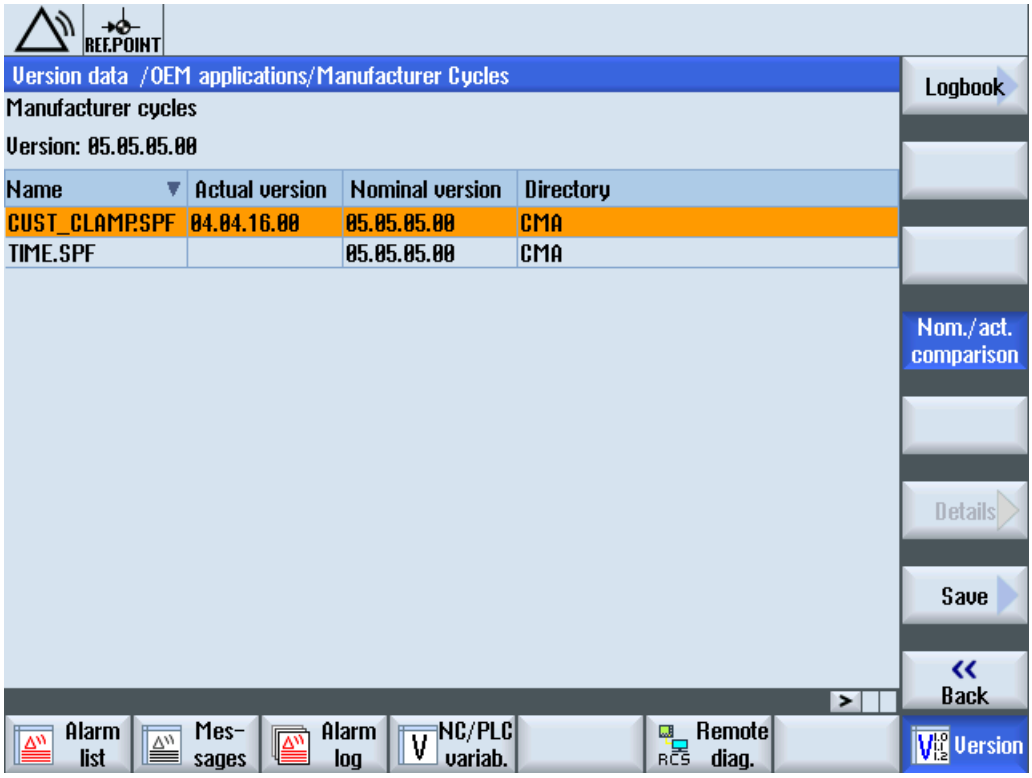
   The "Manufacturer Cycles" window opens.

   The "Planned-actual comparison" softkey is selected by default.

## 7.10.2 Specify cycles version

### General

You have the possibility to compare a required cycles version with the existing cycles version in the version view.



## File overview

The following files are required:

| Name | Path | Meaning |
|---|---|---|
| versions.xml | /siemens/sinumerik/hmi/data/version/oem/cma | Version file |
| versions.xml | /oem/ | Version path file |

## Adapting the version file

The cycles version specification requires a manual adaptation of the version file.

1. Copy the "versions.xml" sample file from the following directory:
   /**siemens**/sinumerik/hmi/data/version/oem/cma
2. Save the sample file in its own directory under /oem.
   The standard directory for this purpose is /oem/sinumerik/hmi/data/version/oem/cma.
3. Open the file.
4. Enter the planned version for the cycles files in the <info> tag with "defaultFileVersion" and the details for the complete version in the <Version> tag.
5. Close the file to save the changes.

**Example:**

```
<info defaultFileType=" *.spf *.cpf" defaultFileVersion="05.05.00.00"
linkname="CMA" linkpath="//NC/CMA.DIR" EffDirOrder="CUS CMA">
<Name>Manufacturer cycles</Name>
<Version>5.5.0.0</Version>
<Link>
<Name>CUS</Name>
<Path>//NC/CUS.DIR</Path>
</Link>
</info>
```

## Adapting the version path file

If the version file is saved in the standard directory, and a current oem\versions.xml is available, no changes are required. In all other cases, the path data can be manually changed:

1. Open the versions.xml file under /oem
2. Replace the standard path in the <Path> tag for the "Manufacturers Cycles" with your own path for the version file.
3. Close the file to save the changes.

**Standard:**

```
<Component>
<Name>Manufacturer Cycles</Name>
<Path>/siemens/sinumerik/hmi/data/version/oem/cma</Path>
```

```
</Component>
```

**Example:**

```
<Component>
<Name>Manufacturer Cycles</Name>
<Path>/oem/sinumerik/hmi/data/version/oem/cma</Path>
</Component>
```

# 7.11 Cycle protection (option)

## 7.11.1 Overview, cycle protection

### Application

With cycle protection, cycles can be encrypted and then stored in the control protected. The processing of cycles with cycle protection in the NC is possible without restrictions.



**Software option**

You require the following software option in order to use this function: "Lock MyCycles".

**Note**

This type of encryption is in compliance with export restrictions and embargo regulations.

In order to protect the manufacturer's know-how, any type of view is inhibited for cycles with cycle protection. If service is required, the machine manufacturer must provide the unencrypted cycle.

**Note**

**End user**

When using encrypted cycles of a machine manufacturer and problems occur, only the service department of the machine manufacturer should be contacted.

**Machine manufacturer**

When using encrypted cycles, the machine manufacturer must ensure that original, unencrypted cycles are archived with the appropriate version management.

**Note**

The cycles are encrypted outside the control with the SinuCom Protector program.

## Encryption

When calling the program, the encrypted NC cycle is generated in the same directory as the original:

- In order to use the more sophisticated encryption, when calling the program, enter option "-e2".
- In order to use encryption compatible with older software versions, when calling the program, enter option "-e1".

The OEM loads the encrypted NC cycle into the control, and activates it using an NC warm restart.

| NOTICE |
| --- |
| **Compatibility of the encryption**<br>We do not recommend loading encrypted cycles into older software versions using option "-e2". |

Siemens cycles that are supplied always match the software release.

Cycles encrypted with "–e1" and "-e2" can be mixed.

## Copying encrypted cycles

The encrypted cycle can be copied and therefore used at other machines.

- **Using encrypted cycles at only one machine**
  If the cycle must not be used at another machine, then the cycle can be permanently linked to a particular machine. The machine data MD18030 $MN_HW_SERIAL_NUMBER can be used for this purpose.
  When the control boots, the unique hardware serial number of the CompactFlash card is saved in this machine data. If a cycle is to be permanently linked to a machine, then in the call header of the cycle, the serial number of the CompactFlash card must be queried (MD18030 $MN_HW_SERIAL_NUMBER). If the cycle identifies a serial number that does not match, then an alarm can be output in the cycle and so prevents further processing. As the code of the cycle is encrypted, there is always a fixed link to a defined hardware.
- **Using encrypted cycles for several machine data**
  If a cycle is to be permanently linked to several, defined machines, then each of the hardware serial numbers must be entered in the cycle. The cycle must be re-encrypted with these hardware serial numbers.

## Handling encrypted cycles

A _CPF file can be deleted or unloaded just like an _SPF or _MPF file. All the encrypted _CPF files are also backed-up when an archive is generated.

- An encrypted cycle cannot be directly selected for processing. It can only be called from a program - or directly in MDI.
- An encrypted cycle cannot be processed with the "Process external" function.

# Spindle function

# 8

## 8.1 Spindle control

You can select from the following spindle control settings:

### End of program

The following M functions are used for the end of program:

| | | |
|---|---|---|
| M function from MD10714 $MN_M_NO_FCT_EOP | Main program end | Spindle continues to run |
| M2, M30 | End of main program and return jump to beginning of program | Spindle stops |
| M17 | End of subprogram and return jump into the main program | Spindle continues to run |

If machine data MD10714 $MN_M_NO_FCT_EOP is not equal to zero, then a distinction is made between M2 / M30 (program end of an NC program) and the M function set in MD10714 $MN_M_NO_FCT_EOP.

### Example

MD10714 $MN_M_NO_FCT_EOP = 32 means "M32" for the program end of a program generated in the "JOG" or "MDA" operating mode.

Among other things, this functionality is required in order to continuously start the spindle in manual operation (e.g. for scratching).

### Configuring keys

If you implement a manual control using the keys on the machine control panel, then this is realized via the following interface signals in the spindle data block:

| | |
|---|---|
| DB3800.DBX5006.0 | Spindle stop |
| DB3800.DBX5006.1 | Spindle start clockwise rotation |
| DB3800.DBX5006.2 | Spindle start, counter-clockwise rotation |

The spindle can be started and stopped in the following state:

| | |
|---|---|
| DB3300.DBX3.7 = 1 | Channel in reset state |
| DB3300.DBX3.6 = 1 | Channel state interrupted |
| DB3300.DBX3.3 = 1 | Program state interrupted |

**Note**

If the spindle must be stopped when the program is running, then set the interface signal DB380x.DBX4.3, "Feed stop/spindle stop" in the user PLC.

Additional settings:

| MD11450 $MN_SEARCH_RUN_MODE | | Search run parameterization |
|---|---|---|
| Bit 1 = 1 | Automatic ASUP start (PROG_EVENT) after output of action sets. Alarm 10208 is not output until the ASUB is completed.<br>Notice:<br>The PROG_EVENT cycle cannot be changed for the SINUMERIK 828D. Manufacturer-specific entries are possible only in the CYCPE_MA and CYCPE_US cycles. | |

## References

For a detailed description to configure the spindle, see

Function Manual, Basic Functions: Spindles (S1)

## Direction of rotation (only for ShopTurn)

In the ShopTurn user interface, ensure that the direction of rotation of the spindle and C axis is correctly displayed and when programming ShopTurn functions, that the correct direction of rotation is executed. You must base these settings on the actual direction of rotation of the spindle/C axis on the machine.

- If you use the functions cylinder surface transformation and face machining, then a precondition is that the function was correctly commissioned.
  See Cylinder surface transformation (TRACYL) (Page 115)
- The direction of rotation of the spindle / C axis for the M function M3 that is displayed in the ShopTurn interface is defined in part using machine data:
  See: Defining the direction of rotation in Section: Setting-up ShopTurn cycles for turning (Page 105)
- The spindle direction of rotation (M3/ M4) is assigned to the positive direction of rotation of the C axis via the interface signal DB38xx.DBX2001.6. The bit defines whether M3 and C + rotate in the same direction (= 0) or in opposite directions (= 1). The corresponding setting options can be found in the following sections:
  - Direction of rotation of counter-spindle (Setting up the counterspindle under ShopTurn (Page 109))
  - Direction of rotation of main spindle (Setting-up ShopTurn cycles for turning (Page 105))

## 8.2 Analog spindle

### Machine axis index for analog spindle

For a spindle which is not assigned to any drive (FD, MSD, SLM), the power display can be controlled via the PLC.

In order that the operating software identifies the spindle as analog spindle, enter the axis index of the analog spindle in the following machine data:

| | |
|---|---|
| MD51068 $MNS_SPIND_DRIVELOAD_FROM_PLC1 | Machine axis index spindle 1<br>Utilization display from the PLC |
| Machine axis index of a spindle (analog) that draws the data for the utilization display in the T,F,S window from the PLC (DB1900.DBB5006). | |

## 8.3 Leadscrew error compensation

Instead of transferring the compensation data via the part program or INI file to the NC, you can enter and check the compensation data for all axes or change the configuration in an input screen.

Finally, the changed data is activated in the NC in one step. The machine data required when activating is set automatically.

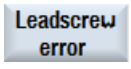
### Procedure







1. Select the "Start-up" operating area and press the "NC" softkey.




2. Press the "Leadscrew error" softkey.

   The "Values of the Compensation Tables" window opens and displays the default compensation tables for an axis or, if no compensation values are available, the measuring system used.

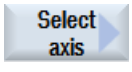






3. Press the "Axis +" or "Axis -" softkey to select the desired axis.

   - OR -




   Press the "Select axis" softkey.

   The "Axis Direct Selection" window opens. Position the cursor on the desired axis and confirm with "OK".






4. Press the "Configuration" softkey to edit the compensation values for a selected axis.
   The "Configuration of the Compensation Tables" window opens.
5. Enter the following:
   - Type of compensation: EEC or CEC.
   - Start position, end position and interpolation point spacing.
   - Only for EEC:
     Used measuring system - if several measuring systems are available, select the desired system from the selection menu.
   - Only for CEC:
     – Compensation table (-): Index of the table in the negative direction.
     – Compensation table (+): Index of the table in the positive direction.




6. Confirm your entries with the "Activate" softkey.
   A dialog box to carry out a NC warm restart is then displayed to activate the compensation. Confirm with "OK".




   The "Values of the Compensation Tables" window opens.
7. Enter the desired compensation values for each interpolation point.




8. Press the "Activate" softkey again to transfer the compensation values to the NC.

### References

Extended Functions (K3) Function Manual

# 8.4 Spindle diagnostics

## 8.4.1 Spindle diagnostics



**Software option**

For the "Spindle diagnostics" you require the "S-Monitor" option.

Main spindle drives are monitored via various status signals. The DRIVE-CLiQ connection at the spindle can be used to evaluate these signals in the drive. The status signals are evaluated for more effective diagnostics of the spindle operating states. The following information is displayed on the user interface with the aid of the "S-Monitor" software option:

- Operating hours
- Temperatures
- Speed/torque
- Clamping system
- Logistics data

## Requirements

- A spindle has an SMI24 sensor module:
  r0459, bit 14 = 1.
- Drive telegram 139 is configured for the spindle.
- The spindle functionality for the machine axis is present, when:
  MD35000 $MA_SPIND_ASSIGN_TO_MACHAX > 0
  The value corresponds to the spindle number.

## References

For further information, please refer to the following documentation:

- Basic Functions (S1), Function Manual; Section "Spindle with SMI24"
- "Start-up a hybrid spindle with SMI24" application description on the Internet at:
  Commissioning of a hybrid spindle with SMI 24 (http://support.automation.siemens.com/), Entry ID 67598396

## Data overview

The "S1 Spindle Diagnostics" window shows the following information:

| Parameter | Value |
|---|---|
| Operating hours<br>• Spindle under control<br>• Spindle under speed | h |
| Number of clamping cycles (tool change)<br>Definition of a clamping cycle: Released clamping state [3] → Clamped clamping state [7, 8 or 10] → Released clamping state [3]. | |
| Motor temperature | °C |
| Clamping state (sensor 1) | V |
| Piston free (sensor 4) | |
| e.g. defined rotary angular position of the shaft (sensor 5) | Yes/No |
| Maximum speed | rpm |

## Procedure

1. Select the "Diagnostics" operating area.
2. Press the menu forward key twice.
3. Press the "Spindle diag." softkey to obtain a data overview.
   The "S1 Spindle Diagnostics" window opens.

**Note**

If several spindles are in operation, select the desired spindle using the "Spindle +" or "Spindle -" softkey.

## 8.4.2 Temperatures

Which temperatures are evaluated and displayed depends on the number and the mounting location (e.g. for sensor S6, bearing temperature front) of the sensors installed in the spindle. The motor temperature is not displayed when the temperature sensor sensor type parameter p0601 = 0 or 1.

**Note**

**Temperature limits**

The limit values displayed here refer to the spindle monitoring and have no effect on the drive parameter values set in the drive that apply for the shutdown response of the spindle. The temperature limit values are preset in the SMI24 and cannot be changed in this window. If necessary, you must adapt the shutdown limit values for the drive.

Up to three temperatures can be evaluated:

| Signal | Sensor | Column/meaning | Description |
|---|---|---|---|
| Motor temperature (Page 202) | KTY | Actual | Current motor temperature. If an overtemperature protection is present and responds, the value 250° C is displayed. |
| Motor temperature Alarm | | Limit | Stored threshold for motor overtemperature as of which an alarm is issued due to KTY motor overtemperature. |
| | | Duration of the upper limit violation | Displays the accumulated hours during which an upper limit violation was present. |
| | | Last upper limit violation | Displays the time stamp [date] and [time] at which the last upper limit violation was present. |
| | | Number of upper limit violations | Accumulated number of upper limit violations. |
| Motor temperature Fault | | Limit | Stored threshold for motor overtemperature as of which a fault is issued due to KTY motor overtemperature. |
| | | Duration of the upper limit violation | Displays the accumulated hours during which an upper limit violation was present. |
| | | Last upper limit violation | Displays the time stamp [date] and [time] at which the last upper limit violation was present. |
| | | Number of upper limit violations | Accumulated number of upper limit violations. |

| Signal | Sensor | Column/meaning | Description |
|---|---|---|---|
| Overtemperature protection | PTC | Actual | When the overtemperature protection responds, "Active" is displayed (in red). The value 250° C is displayed in the "Actual" column as motor temperature. |
| | | Limit | Stored threshold for motor overtemperature as of which a fault is issued due to PTC winding motor overtemperature. |
| | | Last upper limit violation | Displays the time stamp [date] and [time] at which the last upper limit violation was present. |
| | | Number of upper limit violations | Accumulated number of upper limit violations. |
| Additional temperature (Page 202) (e.g. bearing temperature front) | S6 (KTY) | Actual | Current temperature at the measuring point. |
| Additional temperature Alarm | | Limit | Stored temperature threshold as of which an alarm is issued, e.g. due to bearing overtemperature. |
| | | Duration of the upper limit violation | Displays the accumulated hours during which an upper limit violation was present. |
| | | Last upper limit violation | Displays the time stamp [date] and [time] at which the last upper limit violation was present. |
| | | Number of upper limit violations | Accumulated number of upper limit violations. |
| Additional temperature Fault | | Limit | Stored temperature threshold as of which a fault is issued, e.g. due to bearing overtemperature. |
| | | Duration of the upper limit violation | Displays the accumulated hours during which an upper limit violation was present. |
| | | Last upper limit violation | Displays the time stamp [date] and [time] at which the last upper limit violation was present. |
| | | Number of upper limit violations | Accumulated number of upper limit violations. |

## 8.4.3 Motor temperature sensor

### Shutdown response

The alarm and fault thresholds of the motor temperature sensor that are displayed on the user interface as limits, are stored in the SMI24. They are used for the statistical evaluation of the SMI24 and for the shutdown response of the drive. A delay time setting for the violation of the alarm threshold stored in the SMI24 until the fault is triggered is not relevant for the display.

Please note: The motor parameter values of p0604, p0605 (alarm/fault threshold) and the delay time p0606 are predefined and cannot be changed.

The delay time in p0606 has no effect on the values displayed on the user interface. This means: When the alarm threshold is violated and the delay time set in p0606 has expired, the "Motor temperature fault" signal does not change the "Last upper limit violation" display value and the "Number of upper limit violations" is not incremented.

The parameter values of p0604, p0605 (alarm/fault threshold) and p0606 (delay time) have an effect on the shutdown response of the spindle. The spindle is shut down when the fault threshold is violated or the alarm threshold is violated and the delay time set in p0603 has expired.

Currently, the delay time is not predefined by the SMI24. The parameter value of p0603 is set to the factory setting (240 s).

Alarms:

- 207015 = motor temperature sensor alarm
- 207016 = motor temperature sensor fault

## 8.4.4 Additional temperature sensor

### Shutdown response

The alarm and fault thresholds of the additional temperature sensor that are displayed on the user interface as limits, are stored in the SMI24. They are used only for the statistical evaluation of the SMI24. A delay time setting for the violation of the alarm threshold stored in the SMI24 until the fault is triggered is not relevant for the display.

Please note: The parameter values of p4102[0..1] (alarm/fault threshold) and p4103 (delay time) are predefined and cannot be changed.

Changing the parameter values of p4102[0..1] (alarm/fault threshold) and p4103 (delay time) has no effect on the values displayed on the user interface.

The parameter values of p4102[0..1] (alarm/fault threshold) and p4103 (delay time) have an effect on the shutdown response of the spindle. The spindle is shut down when the fault threshold is violated or the alarm threshold is violated and the delay time set in p4103 has expired. These parameters can be adapted accordingly to prevent an unwanted shutdown of the spindle.

Currently, the delay time is not predefined by the SMI24. The parameter value of p4103 is set to the factory setting (0 s). This means: The timer is deactivated.

Alarms:

- 207017 = additional temperature alarm threshold violated
- 207018 = additional temperature fault threshold violated

## 8.4.5 Temperature histograms

The motor temperature and optionally additional temperatures, such as bearing temperature or housing temperature, are displayed in a histogram. The color coding indicates the following temperature ranges:

| Color | Meaning |
|---|---|
| | Temperature range up to the first limit. |
| | Temperature range between the first and second limit. |
| | Temperature range above the second limit. |

The heights of the individual bar fields correspond to the parameterized temperature ranges of the relevant temperature measuring point.

The specified times for the individual temperature ranges of a measuring point (sensor) are the operating hours under speed.

The currently measured temperatures are displayed below the bars.

## 8.4.6 Speed/torque

To check the loads on the spindle, the speed and the torque are recorded in a histogram as a function of the operating hours. The color coding indicates the following states:

- Light green: low load
- Green: rated load
- Yellow: high load
- Orange: critical load

The heights of the individual bars correspond to the percentage time shares (dwell time) in relation to the operating hours under speed. The accumulated number of operating hours under speed for the spindle are shown above or below the respective bar for the corresponding speed or torque range. The width of the bar corresponds to the evaluated speed or torque range.

**Possible conclusions**

- The speed histogram is an indication of the load on the bearings and rotary gland caused by the speed.
- The speed histogram is an indication of the load on the bearings caused by the stock removal forces. The relationship between torque and radial force changes depending on the tool diameter (lever arm), the milling arc (superimposition of the cutting forces of the individual cutting edges) and the cutting force coefficients (the stock removal force consists of cutting force, feedrate force and passive force - only the cutting force is acquired via the torque).

## 8.4.7 Clamping system

The diagnosis of the clamping system offers the following capabilities:

- Fast and robust detection of the clamping states.
- Monitoring of the clamping task and the clamping system.
- Sensors S1 and S4 in the spindle provide information about the clamping device.
- Sensor S5 detects a position, e.g. the defined rotatory angular position of the shaft.

**Clamping states**

The preset limits for the three clamping states are displayed in the upper section of the window.

The clamping state is defined for the specified voltage range of the analog output signal of sensor S1.

The values in the "Voltage min." and "Voltage max." columns indicate in which voltage range the analog signal of sensor S1 must lie for the corresponding clamping state.

The preset speed limitation "Speed max." specifies the maximum permissible speed for the clamping state and is limited to this by the drive.

**Signal overview**

The current state values of the sensors, the tool clamping, the speed setpoint and the active speed limitation are displayed in the signal overview

| Signal | Sensor | Description | Value | Meaning |
|---|---|---|---|---|
| State | S1 | Clamping state | 3 | Released |
| | | | 7 or 8 | Clamped with tool |
| | | | 10 | Clamped without tool |
| Piston free | S4 | Plunger sensing | Yes | Release plunger not in contact with the shaft. |
| | | | No | Release plunger in contact with the shaft. |
| Shaft in the change position | S5 | Position sensing | Yes | For example, shaft is at a defined rotary angular position. |
| | | | No | For example, shaft is not at a defined rotary angular position. |
| Speed setpoint | | Specification of the speed | | Speed specification by the user. |

| Signal | Sensor | Description | Value | Meaning |
|---|---|---|---|---|
| State | | Clamping states | 0 | Sensor S1 not available or state values inactive. |
| | | | 1 | The state initialization is in progress. |
| | | | 2 | Released with signal (error state) |
| | | | 3 | Released |
| | | | 4 | Clamping with tool |
| | | | 5 | Releasing with tool |
| | | | 6 | Releasing without tool |
| | | | 7 | Clamped with tool and S4 = 0 (No). |
| | | | 8 | Clamped with tool and S4 = 1 (Yes). |
| | | | 9 | Clamping without tool |
| | | | 10 | Clamped without tool |
| | | | 11 | Clamped with signal (error state) |

**State bar diagram**

The clamping state is displayed by a state bar. The ranges shown in green indicate the state ranges for "Clamped with or without tool" and "Tool released". The arrow marking indicates the current analog sensor voltage of sensor S1.

## References

For information on the associated interface signals (NC → PLC), refer to the following references:

- NC Variable and Interface Signals List Manual, Section "Axis/spindle-specific signals".

## Procedure

1. The "S1 Spindle Diagnostics" window is open.
   The "S1 Clamping System" window is open without the "S-Monitor" software option.
2. Press the "Clamping system" softkey.
   The "S1 Clamping System" window opens and displays the acquired data.

**Speed limitation**

3. Press the "Speed limitation" softkey to change the limit values for the speeds of the clamping states.
   The "S1 Speed Limitations" window opens.
4. Press the "Change" softkey to enter the desired speed limits depending on the clamping state.
   Confirm with "OK".
   - OR -
3. Press the "Statistics" softkey to read out the clamping times and perform a diagnosis of the clamping system.
   The "S1 Clamping System Statistics" window opens and displays the acquired data.

- OR -

3. Press the "Back" softkey to return to the data overview.

## 8.4.8 Clamping system: Speed limits

For safety reasons and to protect the spindle, only maximum speeds are permitted for certain clamping states. The drive limits the spindle speed in the respective clamping state to the entered speed.

| Clamping state | Meaning |
|---|---|
| Released | The tool clamping system is in contact with the release plunger. Please note: The release plunger presses with the release force on the clamping system. |
| Clamping | The tool clamping system is in the transition state from "Released" to "Clamped with tool". |
| Releasing from state "Clamped with tool" | The tool clamping system is in the transition state from "Clamped with tool" to "Released". |
| Releasing from state "Clamped without tool" | The tool clamping system is in the transition state from "Clamped without tool" to "Released". |
| Clamped with tool | The tool clamping system is in the "Clamped with tool" state. |
| Clamping without tool | The tool clamping system is in the transition state from "Clamped with tool" to "Clamped without tool". |
| Clamped without tool | The tool clamping system is in the "Clamped without tool" state. |

You can adapt the preset speed limits to your requirements by pressing the "Change" softkey.

**NOTICE**

**Damage to the spindle through too high a speed**

Changes to the preset speed limits can result in safety-relevant changes of the operating states and to damage on the spindle.

## 8.4.9 Clamping system: Diagnostic statistics

This window displays the statistics for clamping operations and speed violations.

| Statistical data | Meaning |
|---|---|
| Number of correct clamping cycles | Accumulated number of correct clamping cycles with or without tool.<br>Correct clamping cycle means: "Released" clamping state [3] → "Clamped" clamping state [7, 8 or 10] → "Released" clamping state [3]. A clamping cycle is considered to be correct when a clamping and subsequent release operation are performed via the appropriate clamping states (without "State return"). Starting from one clamping state, the next clamping state that can be reached must not be the previous state. There is no dependency on time. |
| Clamping time (without tool) | The time within a clamping operation between the clamping states "Tool released" and "Clamped without tool". |
| • Within tolerance | Accumulated number of clamping operations without tool that were within the specified time. |
| • Outside tolerance | Accumulated number of clamping operations without tool that were outside the specified time. |
| • Specification | Factory specification of the time for the clamping operation without tool. |
| • Current | Measured time for the last performed clamping operation without tool. |
| Clamped (without tool) | Diagnostics during the "Clamped without tool" clamping state. |
| • Speed violations | Accumulated number of events during which the spindle speed was greater than the set limit as delivered. |
| • Speed threshold | Limit for the spindle speed in the "Clamped without tool" clamping state (as-delivered value).<br>Changing this speed limit via the diagnostics overview "S1 clamping system" → "Speed limits" → "Change" has no effect on the value specified here. |
| Faulty clamping/releasing operations (total) | Accumulated number of clamping operations that have not run through the clamping states "Released" clamping state [3] → "Clamped" clamping state [7, 8 or 10] → "Released" clamping state [3] without interruption.<br>A clamping cycle is considered to be faulty when, starting from one state, the next state that is reached is the previous state again ("State return"). |

## 8.4.10 Clamping system: Clamping time statistics

This window displays the statistics for the determined clamping times.

| Statistical data | Meaning |
|---|---|
| Last determined clamping time | Measured time for the last performed clamping operation.<br>Definition of a clamping operation: "Released" clamping state [3] → "Clamped with tool" clamping state [7 or 8]. Clamping state 3 = Released; clamping state 7 = Clamped with tool and S4 = 0 (No); clamping state 8 = Clamped with tool and S4 = 1 (Yes) |
| Reference clamping time (with tool) | During the spindle start-up phase, the first 100 clamping operations are measured by the SMI24 and the minimum value, maximum value and the mean value determined. |
| • Minimum | The shortest clamping time determined from the first maximum 100 clamping operations. |
| • Maximum | The longest clamping time determined from the first maximum 100 clamping operations. |
| • Mean | The mean clamping time determined from the first maximum 100 clamping operations. |
| Mean clamping time (without tool) | After the first 100 clamping times determined for the reference time, the mean values for the other measured clamping times are calculated. |
| • 24 h | Mean clamping time of the clamping cycles determined in the last 24 hours. |
| • 10 days | Mean clamping time of the clamping cycles determined in the last 10 days. |
| • 100 days | Mean clamping time of the clamping cycles determined in the last 100 days. |

If a Power On of the SMI24 is performed during the start-up phase, the previously determined values are discarded. The counter for the first 100 clamping operations is set to zero and the reference times up to the 100th clamping operation can be determined again. After the start-up phase (first 100 clamping operations), the values of the reference clamping time are "frozen". Even after a Power On of the SMI24, the values remain stored and can no longer be changed.

## 8.4.11 Fetching the logistics data

### Requirement

The spindle characteristics can only be read out with the "Service" access level.

### Spindle characteristics

The following spindle manufacturer characteristics can be read out via DRIVE-CLiQ:

- Manufacturer's identifier
- Drawing number

- Serial number
- Date of manufacture
- Service date and service information (1...4):
  The service information corresponds to the reference number for the service report from the WEISS Spindeltechnologie GmbH company.

# Tool management

# 9

## 9.1 Machine data for tool management

### 9.1.1 Settings with/without magazine management

#### Setting without magazine management

Make the following settings if you are commissioning the tool management **without** NC magazine management:

| MD18080 $MN_MM_TOOL_MANAGEMENT_MASK | Memory for the tool management is reserved step-by-step (SRAM). |
|---|---|
| = 02H | |
| Bit 1 | Provide monitoring data. |

| MD20310 $MC_TOOL_MANAGEMENT_MASK | Activating the tool management functions. |
|---|---|
| = 02H | |
| Bit 1 | Monitoring functions of tool management active. |

| MD17530 $MN_TOOL_DATA_CHANGE_COUNTER | Marking the changed tool data for HMI. |
|---|---|
| = 1FH | |
| Bit 0 | Tool status change. |
| Bit 1 | Tool remaining unit quantity change. |
| Bit 2 | Tool data change service. |
| Bit 3 | Magazine data in the change service. |
| Bit 4 | ISO data in the change service. |

| MD28450 $MC_MM_TOOL_DATA_CHG_BUFF_SIZE | Buffer for changing tool data (DRAM). |
|---|---|
| = 100 | Number of entries in the buffer for the OPI change service for tool data. Dynamic memory is used. This buffer is only created if either bit 2 or bit 3 is set in MD17530 $MN_TOOL_DATA_CHANGE_COUNTER. |

| MD19320 $ON_TECHNO_FUNCTION_MASK | |
|---|---|
| =2000000H | |
| Bit 25 | Enables the tool monitoring function without magazine management. |

### Setting with magazine management

Make the following settings if you are commissioning the tool management **with** NC magazine management:

| MD18080 $MN_MM_TOOL_MANAGEMENT_MASK | Memory for the tool management is reserved step-by-step (SRAM). |
|---|---|
| = 03H | |
| Bit 0 | Provide tool management data. |
| Bit 1 | Provide monitoring data. |

| MD20310 $MC_TOOL_MANAGEMENT_MASK | Activating the tool management functions. |
|---|---|
| = 03H | |
| Bit 0 | Magazine management active. |
| Bit 1 | Monitoring functions of tool management active. |

| MD17530 $MN_TOOL_DATA_CHANGE_COUNTER | Marking the changed tool data for HMI. |
|---|---|
| = 1FH | |
| Bit 0 | Tool status change. |
| Bit 1 | Tool remaining unit quantity change. |
| Bit 2 | Tool data change service. |
| Bit 3 | Magazine data in the change service. |
| Bit 4 | ISO data in the change service. |

| MD28450 $MC_MM_TOOL_DATA_CHG_BUFF_SIZE | Buffer for changing tool data (DRAM). |
|---|---|
| = 100 | Number of entries in the buffer for the OPI change service for tool data. Dynamic memory is used. This buffer is only created if either bit 2 or bit 3 is set in MD17530 $MN_TOOL_DATA_CHANGE_COUNTER. |

| MD19320 $ON_TECHNO_FUNCTION_MASK | |
|---|---|
| = 10H | |
| Bit 4 | Enables the tool monitoring function with magazine management. |

## 9.1.2 Configuring the access levels of the tool management

### Access levels for selected parameters

| MD51199 $MNS_ACCESS_WRITE_TM_GRIND | Access level, write TM grinding data. |
|---|---|
| MD51200 $MNS_ACCESS_WRITE_TM_GEO | Access level, write TM geometry data. |
| MD51201 $MNS_ACCESS_WRITE_TM_WEAR | Access level, write TM wear data. |
| MD51202 $MNS_ACCESS_WRITE_TM_WEAR_DELTA | Access level, restricted writing of tool wear values. |

| | |
|---|---|
| MD51203 $MNS_ACCESS_WRITE_TM_SC | Access level, write TM additive offsets. |
| MD51204 $MNS_ACCESS_WRITE_TM_EC | Access level, write TM operating offsets. |
| MD51205 $MNS_ACCESS_WRITE_TM_SUPVIS | Access level, write TM monitoring data. |
| MD51206 $MNS_ACCESS_WRITE_TM_ASSDNO | Access level, write TM unique D number. |
| MD51207 $MNS_ACCESS_WRITE_TM_WGROUP | Access level, write TM wear groups (magazine location / magazine). |
| MD51208 $MNS_ACCESS_WRITE_TM_ADAPT | Access level, write TM tool adapter geometry data. |
| MD51209 $MNS_ACCESS_WRITE_TM_NAME | Access level, write TM tool name and duplo. |
| MD51210 $MNS_ACCESS_WRITE_TM_TYPE | Access level, write TM tool type. |

The access level of individual parameters can be overlaid through settings in the <PARAMETERCONFIGURATION> tag, "Access level" entry (see SectionConfigure the list parameters (Page 237)).

## Access levels for the "Further Details" window

| | |
|---|---|
| MD51198 $MNS_ACCESS_READ_TM_ALL_PARAM | Protection level, TM details - read all parameters.<br>Depending on the machine data, the "Further details" softkey is either hidden or displayed. |
| MD51215 $MNS_ACCESS_WRITE_TM_ALL_PARAM | Access level, TM details - write all parameters. |

## Access levels for certain functions of the tool management

| | |
|---|---|
| MD51216 $MNS_ACCESS_TM_TOOL_CREATE | Access level, TM, create tool. |
| MD51217 $MNS_ACCESS_TM_TOOL_DELETE | Access level, TM, delete tool. |
| MD51218 $MNS_ACCESS_TM_TOOL_LOAD | Access level, TM, load tool. |
| MD51219 $MNS_ACCESS_TM_TOOL_UNLOAD | Access level, TM, unload tool. |
| MD51220 $MNS_ACCESS_TM_TOOL_MOVE | Access level, TM, move tool. |
| MD51221 $MNS_ACCESS_TM_TOOL_REACTIVATE | Access level, TM, reactivate tool. |
| MD51222 $MNS_ACCESS_TM_TOOL_MEASURE | Access level, TM, measure tool. |
| MD51223 $MNS_ACCESS_TM_TOOLEDGE_CREATE | Access level, TM, create tool cutting edge. |
| MD51224 $MNS_ACCESS_TM_TOOLEDGE_DELETE | Access level, TM, delete tool cutting edge. |
| MD51225 $MNS_ACCESS_TM_MAGAZINE_POS | Access level, TM, position magazine. |

## 9.1.3 Additional settings

### Additional settings

You can enable further functions on the user interface via the following machine data / setting data:

| MD52270 $MCS_TM_FUNCTION_MASK | |
|---|---|
| Bit 0 | Creating a tool at the magazine location is not permitted. |
| Bit 1 | Load/unload lock if the machine is not in the reset state. |
| Bit 2 | Load/unload lock for EMERGENCY OFF. |
| Bit 3 | Loading/unloading or relocating tool in/out of spindle locked. |
| Bit 4 | Tool loaded directly into spindle. |
| Bit 5 | Using the grinding configuration file. |
| Bit 6 | Relocating a tool in/out of spindle is permitted despite lock (see bit 3). |
| Bit 7 | Tool is created using the number. |
| Bit 8 | Hide tool relocation. |
| Bit 9 | Hide magazine positioning. |
| Bit 10 | Reactivate tool with magazine positioning. |
| Bit 11 | Reactivate tool in all monitoring types. |
| Bit 12 | Hide reactivating tool. |

| SD54215 $SNS_TM_FUNCTION_MASK_SET | |
|---|---|
| Bit 0 | Diameter display for rotating tools.<br>Cutting edge parameters 6 (tool nose radius) and 15 (wear radius) are not displayed as a radius value, but as a diameter value for the following tool types: Types 100 to 299, 580, 710, 711, 712, 713, 714 and 725.<br>Cutting edge parameter 7 (outside radius) is not displayed as a radius value, but as a diameter value for the following tool types: Types 140 and 714. |
| Bit 1 | M4 is the standard direction of rotation for all turning tools. When creating turning tools, the direction of rotation is pre-assigned with M4. |
| Bit 2 | A name is not suggested when creating a tool. |
| Bit 3 | Input lock, tool name and tool type for loaded tools. For loaded tools, the tool name and the tool type can no longer be changed. |
| Bit 4 | Input lock for loaded tools if the channel is not in the reset state. |
| Bit 5 | Tool wear entries are added: Wear data is entered in addition to the already existing wear value. |
| Bit 6 | Only numerical entries are permitted in the "Tool identifier". |
| Bit 7 | Hide tool monitoring parameters. The tool monitoring parameters are hidden in the user interface. |
| Bit 8 | Diameter display for face axis - geometry. The geometry value of the transverse axis is displayed as a diameter value, if a transverse axis is defined in MD20100 $DIAMETER_AX_DEF and SD42940 $TOOL_LENGTH_CONST is set to 18 and SD42950 $TOOL_LENGTH_TYPE is set to 2. |

| | |
|---|---|
| Bit 9 | Diameter display for face axis - wear. The wear value of the transverse axis is displayed as a diameter value, if a transverse axis is defined in MD20100 $DIAMETER_AX_DEF and SD42940 $TOOL_LENGTH_CONST is set to 18 and SD42950 $TOOL_LENGTH_TYPE is set to 2. |
| Bit 10 | Enable tool load/relocate to buffer storage locations. The magazine number can be entered into the load dialog box. It is therefore possible to access the buffer storage via magazine number 9998. |
| Bit 11 | Creating new tools at gripper locations is blocked. |
| Bit 12 | Measuring tools are not unloaded with the "Unload all" function. |

## Dependencies

The setting data SD54215 $TM_FUNCTION_MASK_SET and the machine data MD20360 $TOOL_PARAMETER_DEF_MASK have the following dependencies:

| Dependency | | | Description | Recommendation |
|---|---|---|---|---|
| **SD54215** | **⇔** | **MD20360** | | |
| Bit 9 | ⇔ | Bit 0 | If bit 0 is set in MD20360, then SD54215 bit 9 has no effect because the NC already returns the linear wear value of the transverse axis as a diameter value. | It is recommended that the setting MD20360 bit 0 and bit 1 not be used.<br>**Instead, use SD4215 bit 9 and bit 8**.<br>MD20360 bit 0 and bit 1 are effective only for turning and grinding tools. This setting therefore cannot be used on a turning/milling machine because the linear component of the transverse axis applies for turning tools in the diameter and for milling tools in the radius.<br>When the setting data is used, only the display goes to diameter, the internal value in the NC remains at radius. |
| Bit 8 | ⇔ | Bit 1 | If bit 1 is set in MD20360, then SD54215 bit 8 has no effect because the NC already returns the linear geometry value of the transverse axis as a diameter value. | |
| Bit 0 | ⇔ | Bit 11 | If bit 11 is set in MD20360, then SD54215 bit 0 has no effect with regard to cutting edge parameter 6 (tool nose radius) because the NC already returns the tool nose radius as a diameter value. | It is recommended that the setting MD20360 bit 11 and bit 12 not be used.<br>**Instead, set bit 0 in SD54215.**<br>MD20360 bit 11 and bit 12 apply to all tool types, i.e. also to turning tools. This setting therefore cannot be used on a turning/milling machine because the tool nose radius of a turning tool should never be specified in the diameter.<br>When the setting data is used, only the display goes to diameter, the internal value in the NC remains at radius. |
| Bit 0 | ⇔ | Bit 12 | If bit 12 is set in MD20360, then SD54215 bit 0 has no effect with regard to cutting edge parameter 15 (wear radius) because the NC already returns the wear radius as a diameter value. | |

## Enabling the graphic tool and magazine display function

Make the following settings when the tools and the magazine should be represented graphically:

| MD52271 $MCS_TM_MAG_PLACE_DISTANCE | | Distance between individual magazine locations in the unit of measurement of the basic NC system. |
|---|---|---|
| > 0 | The tools and magazine locations are displayed graphically. | |
| = 0 | The tools and magazine locations are not displayed graphically. | |

You must make the following settings on the user interface:

1. Press the "Settings" softkey in the "Tool list" window.
   The "Settings" window opens.
2. Activate the "Switch on graphical magazine display" checkbox.
   This checkbox is offered only when the MD52271 $MCS_TM_MAG_PLACE_DISTANCE machine data has a value > 0.

Supplementary conditions:

- MD52200 $MCS_TECHNOLOGY = 2 (milling)
- MD52201 $MCS_TECHNOLOGY_EXTENSION = 0

## Default settings for unloading/loading and relocating

| MD52272 $MCS_TM_TOOL_LOAD_DEFAULT_MAG | Default magazine for loading tools. |
|---|---|
| = 0 | No default magazine, default value |

| MD52273 $MCS_TM_TOOL_MOVE_DEFAULT_MAG | Default magazine for relocating tools. |
|---|---|
| = 0 | No default magazine, default value |

| MD52274 $MCS_TM_TOOL_LOAD_STATION | Number of loading station. |
|---|---|
| = 0 | All configured stations are considered, default value |

## Input limits for wear input

| | |
|---|---|
| MD51212 $MNS_TM_WRITE_WEAR_ABS_LIMIT | Maximum value of the tool wear. |
| MD51213 $MNS_TM_WRITE_WEAR_DELTA_LIMIT | Maximum differential value of restricted tool wear input. |
| MD51214 $MNS_TM_WRITE_LIMIT_MASK | Range of validity of the restricted tool wear input. |

## Setting the manual tool function

| MD22562 $MC_TOOL_CHANGE_ERROR_MODE | Response when errors occur at tool change. |
|---|---|
| Bit 1 = 1 | Manual tools can be loaded. |
| Bit 2 = 1 | Active D No. > 0 and active T No. = 0 generate an alarm message.<br>Active D No. > 0 and active D No. = 0 generate an alarm message. |

**Note**

If, in machine data MD22562 $MC_TOOL_CHANGE_ERROR_MODE bit 2 = 1, then you have the option, using a correction selection (D No.) in the program, to stop the machine until the PLC signals that the tool change has been completed. To do this, machine data MD20270 $MC_CUTTING_EDGE_DEFAULT must be set to = 0.

## 9.2 Configuring the user interface

### Configurable settings

The following settings can be configured:

- General settings
- Configure the tool lists
- Configure the list parameters
- Configure the tool types
- Configure the "Additional Data" window
- Configure the "New Tool - Favorites" window
- Configure the "New Tool" window
- Configure the default values for new tools
- Configure the "Details" window
- Assign names for magazines and magazine locations
- Assign names for magazine location types
- Assign magazines to channels
- Coolant and tool-specific functions
- Reason for the tool change when reactivating
- Configure the code carrier connection

The following sections explain how you configure these adaptations.

### Requirement for grinding technology

Set the "Universal" technology and set:
MD52270 $MN_TM_FUNCTION_MASK bit 5 = 1.

Only the grinding tool types can be then selected on the user interface.

## Procedure

1. Open the directory: /**siemens**/sinumerik/hmi/template/cfg
2. Copy the "oem_sltmlistconfig.xml" template file

   **Note**

   **Other template for the grinding technology**

   The "sltmaddgrindinglistconfig.xml" template expands a turning configuration by adding grinding tools. Copy the file from the directory: /**siemens**/sinumerik/hmi/template/cfg

3. Copy the file to the directory /**oem**/sinumerik/hmi/cfg
   OR: /**user**/sinumerik/hmi/cfg
4. Change the file name according to the set technology:

| File name | Technology |
|---|---|
| sltmlistconfig.xml | Milling technology |
| sltmturninglistconfig.xml | Turning technology |
| sltmplclistconfig.xml | PLC tool management "TRANSLINE 2000" |
| sltmcirculargrindinglistconfig.xml | Cylindrical grinding technology |
| sltmsurfacegrindinglistconfig.xml | Surface grinding technology |

5. Remove the comments of the example in the template to create your specific configuration.

## Configuration examples

Examples for all possible configurations are provided in the template oem_sltmlistconfig.xml, which are described individually in the following sections:

```
 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
- <CONFIGURATION>
<!--  ************************************************************-->
<!--  enter your configuration behind this comment -->
<!--  ************************************************************ -->


 Configuration examples


<!--  ************************************************************ -->
<!--  enter your configuration ahead this comment -->
<!--  ************************************************************ -->
 </CONFIGURATION>
```

Enter your configuration using the following xml identifiers:

| Identifier <tag> | for the following setting: |
|---|---|
| <SETTINGS> | General settings (Page 219) |
| <LISTCONFIGURATION> | Configure the tool lists (Page 224) |
| <PARAMETERCONFIGURATION> | Configure the list parameters (Page 237) |
| <TOOLTYPECONFIGURATION> | Configure tool types (Page 243) |

| Identifier <tag> | for the following setting: |
|---|---|
| <MOREDATACONFIGURATION> | Configuring the "More data" window (Page 244) |
| <NEWTOOLFAVORITECONFIGURATION> | Configure the "New tool - favorites" window (Page 246) |
| <NEWTOOLCONFIGURATION> | Configuring the "New Tool" window (Page 246) |
| <NEWTOOLDEFAULTVALUECONFIGURATION> | Configuring default values for new tools (Page 247) |
| <DETAILSCONFIGURATION> | Configuring the "Details" window (Page 251) |
| <MAGAZINEPLACENAMECONFIGURATION> | Assigning a name for magazines and magazine locations (Page 254) |
| <PLACETYPECONFIGURATION> | Assigning a name for magazine location types (Page 256) |
| <CHANNELMAGAZINEASSIGNMENT> | Assignment of magazines to channels (Page 257) |
| <TOOLCHANGEREASON> | Reason for the tool change when reactivating (Page 262) |
| <IDENTCONNECTIONCONFIGURATION> | Configure the code carrier connection (Page 265) |

## 9.2.1 General settings

### <SETTINGS> tag

The general settings for the tool management user interface are stored in the <SETTINGS> tag.

| Entries | Meaning |
|---|---|
| ReactivateWithMagPos | True - reactivate with positioning of the tool to the tool magazine loading point.<br>False - default setting, reactivate without positioning the tool magazine. |
| ReactivateAllMonitorModes | True - the actual values of all the monitoring types set in the NC are reset.<br>False - default setting, actual value of the active monitoring type is reset. |
| ReactivateEnabled | True - default setting, reactivation is enabled.<br>False - reactivation is disabled. |
| CreateNewToolDialog | True - the "New Tool" window is displayed. In this window, you can specify the tool name and the size of the tool (oversized tool) before actually creating the tool. Other input options can be defined by configuring the "New tool" window.<br>False - default setting, the "New Tool" window is not displayed. Once the name has been entered, the tool will be created directly in the tool list. |

| Entries | Meaning |
|---|---|
| CreateNewMultiToolDialog | True - default setting, the "New Multitool" window is displayed. In this window, you can configure the following before actually creating the multitool:<br>• Multitool names<br>• Number of multitool locations<br>• Type of clearance input<br>• Value of the location clearance for each multitool location<br>False - the "New Multitool" window is not displayed. Once the name has been entered, the multitool will be created directly in the tool list. |
| MagazineSelectionDialog | True - a dialog box displaying the buffer, tool magazine and NC memory is displayed. A mark can be added to or removed from each of these elements to indicate whether that element appears in the list display. The "Go to" softkey can be used to select an element directly in the list display.<br>False - no dialog box for magazine selection. The system switches between the buffer, the tool magazines and the NC memory.<br>If there is no MagazineSelectionDialog entry in the <Settings> tag, the dialog box is automatically displayed with more than one tool magazine (excluding the system magazine) when magazine configuration is taking place. If there is only one tool magazine, the switchover function is used. |
| NewToolFavoritesOnly | Refers to the "New tool" function.<br>True - only those tool types that are configured as favorites are listed.<br>False - default setting, all tool types are listed. |
| SortationInFirstEtcLevel | True - the "Sorting" function is placed at the first softkey level. The "Continue" function is omitted.<br>False - default setting, sorting at all softkey levels. |
| ToolBufferOnceOnTop | True - default setting, buffer is displayed at the start of the list.<br>False - buffer is displayed for each magazine. Only those locations are displayed that are assigned to the magazine. |
| UnloadToolFromLoadPlaceEnabled | True - a tool at the load position can be removed from the load position using the "Unload" softkey.<br>False - default setting. The "Unload" softkey is not displayed for a tool at the load position. |
| AutoHNumberDisplayEnabled | Column of the cutting parameter "IsoHNoDPH - ISO H number" depending on the machine configuration.<br>True - default setting, the column is automatically displayed.<br>False - the column is not automatically displayed. |
| AutoUniqueDNumberDisplayEnabled | Column of the cutting edge parameter "UniqueDNo" depending on the machine configuration.<br>True - default setting, the column is automatically displayed.<br>False - the column is not automatically displayed. |

| Entries | Meaning |
|---|---|
| AccessLevelDeleteAllDNos | The following entry specifies the access level as of which the "Delete all D numbers delete" function is offered.<br>Default setting: Access level 3 (user) - relevant only when the "Unique D number" setting has been made. |
| MagazineNumberOfDCheckAndActivate | Magazine number to which the D check (check for unique D number) and Activate (activation of a wear group) are to be applied.<br>Default setting: -2<br>Significance: All magazines that are connected to a toolholder or a spindle. |
| AutoFixedPlaceDisplayEnabled | Column of the tool parameter "ToolStateFixed" - tool status - tool is fixed-location coded and acts dependent on the magazine configuration.<br>True - default setting, the column is automatically hidden, if all of the configured magazines are fixed location coded.<br>False - the column is not automatically hidden. |
| MagPlaceAdapterDisplayEnabled | Columns of the magazine location adapter parameters are displayed in the "Magazine" window.<br>False - default setting, parameters are not displayed. |
| AdaptTransDataViewEnabled | The magazine location adapter transformed display of the cutting edge parameters is enabled. It can be switched on and off in the settings dialog of the tool lists.<br>False - default setting, the magazine location adapter transformed display of the cutting edge parameters is not enabled. |
| ActiveCompensationsEnabled | The "Working offsets" function is enabled. This function depends on the machine configuration. Unique D number, magazine location adapter and magazine management must be set.<br>False - default setting, the "Working offsets" function is not enabled. |
| M2NLockEnabled | Locking of an M to N switchover while a function of the tool management is being executed.<br>True - default setting, switchover is locked. |
| AccessLevelWriteDetailsAllParameter | The following entry specifies the access level as of which write access applies in the "Details - All Parameters" window.<br>Default setting: Access level 7 (keyswitch position 0) - can be overlaid through the setting in MD51215. |
| AccessLevelReadDetailsAllParameter | The following entry specifies the access level as of which the "Further details" function is offered in the "Details" dialog.<br>Default setting: Access level 7 (keyswitch position 0) - can be overlaid through the setting in MD51198. |
| FixNumberOfMultiToolPlaces | Multitools are created with a fixed number of multitool locations. There is no prompt when you create the multitool. |
| FixKindOfDistInMultiTools | Multitools are created with a fixed type of clearance input of the multitool locations. There is no prompt when you create the multitool.<br>Possible input value: 3 - clearance of the multitool locations as an angle. |

| Entries | Meaning |
|---|---|
| DlDisplayMode | • DisplayDistributed - default setting, the setup offsets EC are displayed in the tool list and the additive offsets SC are displayed in the tool wear list.<br>• EcInWearList - the setup offsets EC are displayed in the tool wear list and the additive offsets SC are not displayed. |
| CombinedToolMonitoring | Specifies the combined monitoring types offered in the toggle field:<br>• Time/Count - tool life and workpiece count<br>• Time/Wear - tool life and wear monitoring<br>• Count/Wear - workpiece count and wear monitoring<br>• Time/Count/Wear - tool life, workpiece count and wear monitoring |
| FiltersCombinedWithOr | Defines the filter criteria, which are linked in the filter window using a logical OR. The default setting for all the filter criteria is a logical AND. The possible entries are:<br>• "StandBy" - for tools that are ready for use<br>• "Active" - for tools with active identifier<br>• "Locked" - for locked tools<br>• "Prewarn" - for the prewarning limit<br>• "RemainingPieces" - for the remaining unit quantity<br>• "RemainingTime" - for the remaining tool time<br>• "MarkedToUnload" - for the unload identifier<br>• "MarkedToLoad" - for the load identifier |
| MagazineMoveMessage | True - "Magazine movement running" or "Magazine movement ended" message is displayed.<br>False - default setting, message is not displayed. |
| ToolLoadErrorMessage | Refers to the output of the fault message due to a negative PLC.<br>True - default setting, the fault message is displayed.<br>False - the fault message is not displayed. |

The following settings refer to markings in the lists:

True - marking is displayed.

False - marking is not displayed (default setting).

If several markings apply to one location, the most important information is displayed. For instance, if a tool is locked, and at the same time, the "Tool position" and the "Reference

direction holder angle" for this tool do not match, then a red cross is displayed for locked. The yellow triangle for the conflict cannot be seen.

| Entries | Meaning |
|---|---|
| ShowSymbolActivePlace | Marking of the magazine location at the machining position.<br>True - default setting applies for the "Revolver" magazine type.<br>False - setting for all other magazine types. |
| ShowSymbolActiveMultiToolPlace | Marking of the multitool location at the machining position. Is only displayed when the multitool is located in the spindle or the toolholder.<br>True - default setting. |
| ShowSymbolLoadPlace | Selection of the magazine location at the loading position.<br>True - default setting applies for all magazine types (except Revolver is true).<br>False - setting for the "Revolver" magazine type. |
| ShowSymbolChangeInSpindlePlace | Marking of the magazine location at the change position.<br>True - default setting applies for all magazine types (except Revolver is true).<br>False - setting for the "Revolver" magazine type. |
| ShowSymbolActiveTool | Marking of the active tool.<br>False - default setting. |
| ShowSymbolProgrammedTool | Marking of the follow-on tool.<br>True - default setting. |
| ShowSymbolActiveDNo | Marking of the active cutting edge.<br>False - default setting. |
| ShowSymbolActiveDNoInTool | Marking of the active tool by marking the active cutting edge.<br>True - default setting for configuration without magazine management.<br>False - setting for configuration with magazine management. |
| ShowSymbolAcitveDlNo | Marking of the active DL number.<br>True - default setting. |
| ShowConflictPositionAndDirection | Marking of a conflict; tool position and cutting direction do not match.<br>True - default setting. |
| ShowConflictActiveToolWarning | Marking of a conflict; revolver magazine type and the active tool are not at the the machining position.<br>True - default setting. |
| ShowSymbolActiveIsoHNoL1 | Marking of the active H number for length 1 in the ISO tool list.<br>true: Default setting. |
| ShowSymbolActiveIsoHNoL2 | Marking of the active H number for length 2 in the ISO tool list.<br>True - default setting. |
| ShowSymbolActiveIsoHNoL3 | Marking of the active H number for length 3 in the ISO tool list.<br>True - default setting. |

| Entries | Meaning |
|---|---|
| ShowSymbolActiveIsoDNo | Marking of the active D number in the ISO tool list.<br>True - default setting. |
| ShowSymbolActiveIsoHDNo | Marking of the active HD number in the ISO tool list.<br>True - default setting. |

### Example

```
<SETTINGS>
     <ReactivateWithMagPos value="false" type="bool" />
     <MagazineMoveMessage value="false" type="bool" />
     <CreateNewToolDialog value="true" type="bool" />
     <MagazineSelectionDialog value="false" type="bool" />
     <AccessLevelWriteDetailsAllParameter value="4" type="int" />
</SETTINGS>
```

## 9.2.2 Configure the tool lists

### <LISTCONFIGURATION> tag

The entries used for configuring the following list views are located in the <LISTCONFIGURATION> tag:

- "Tool List" window, identifier <SlTmToollistForm>, see the 1st example.
- "Tool Wear" window, <SlTmToolwearForm> identifier
- "Magazine" window, <SlTmToolmagazinForm> identifier
- "Tool Data OEM" window, <SlTmTooloemForm>identifier
  This window is hidden in the default configuration. The window is only displayed when you enable it with <Enabled>, see the 3rd example.
- "Tool Selection" window, <SlTmToolselectionForm> identifier: In order to select a tool this window can be displayed in the program input screen.

| Entry | Meaning |
|---|---|
| Enabled | True - window is displayed.<br>False - window is hidden. |
| CAPTION | Definition of the list name.<br>See 1st example: "Tool list" appears on the user interface for the "TM_CAPTION_TOOLLIST" identifier. |
| COLUMNX | Column selection: X stands for the column number. |
| TOOLTYPE_ALL | The column setting applies to all tool types. |
| TOOLTYPE_XXX | The column setting applies to a particular tool type. XXX stands for the number of the tool type. |

| Entry | Meaning |
|---|---|
| EMPTY_MAG_PLACE | The column setting applies to an empty magazine location. Empty magazine locations appear in a list sorted according to magazine. |
| Item | Identifier of the list parameter.<br>See example: The column in which the magazine number and magazine location are displayed is created for the "ToolInMagInPlace" identifier.<br>Tool parameter identifiers (Page 228)<br>Cutting parameter identifiers (Page 230)<br>Monitoring parameter identifiers (Page 232)<br>Magazine location parameter identifiers (Page 234) |
| DELETE_COLUMN | Delete column. |
| INSERT_COLUMN | Insert column. |

**Note**

Independent of their programming, the entries are effective in the following sequence:

1. COLUMNX
2. DELETE_COLUMN
3. INSERT_COLUMN

**Note**

The Enabled entry does not apply for the "Tool Selection" window. The other entries apply.

## Example 1: Configuring the magazine list - magazine location and tool location type

**Note**

The columns of the list configuration cannot be taken from the display in the user interface in all cases because some columns are automatically inserted and therefore not included in the count of the configuration.

## Variant 1

Column 10 and column 11 are overwritten. If column 10 and column 11 do not exist, the new columns will be added to the existing columns.

```
<LISTCONFIGURATION>
    <!-- **** Magazine list *** -->
    <SlTmToolmagazinForm>
      <!-- **** Column 10 *** -->
      <COLUMN10>
        <!-- **** For all tool types *** -->
        <TOOLTYPE_ALL>
```

```
        <!-- **** Display magazine location type *** -->
        <Item value="MagPlaceType" type="QString" />
      </TOOLTYPE_ALL>
    </COLUMN10>
    <!-- **** Column 11 *** -->
    <COLUMN11>
      <!-- **** For all tool types *** -->
      <TOOLTYPE_ALL>
        <!-- **** Display tool location type *** -->
        <Item value="ToolPlaceSpec" type="QString" />
      </TOOLTYPE_ALL>
      <!-- **** For all multitools *** -->
      <TOOLTYPE_9997>
        <!-- **** Display for all multitool location types *** -->
        <Item value="MultiToolPlaceSpec" type="QString" />
      </TOOLTYPE_9997>
    </COLUMN11>
  </SlTmToolmagazinForm>
</LISTCONFIGURATION>
```

## Variant 2

The new columns are inserted in the standard configuration between the "Magazine location locked" and "Oversized tool" columns.

```
<LISTCONFIGURATION>
  <!-- **** Magazine list *** -->
  <SlTmToolmagazinForm>
    <!-- **** Insert column *** -->
    <INSERT_COLUMN>
      <!-- **** Column 7 *** -->
      <COLUMN7>
        <!-- **** For all tool types *** -->
        <TOOLTYPE_ALL>
          <!-- **** Display magazine location type *** -->
          <Item value="MagPlaceType" type="QString" />
        </TOOLTYPE_ALL>
      </COLUMN7>
      <!-- **** Column 8 *** -->
      <COLUMN8>
        <!-- **** For all tool types *** -->
        <TOOLTYPE_ALL>
          <!-- **** Display tool location type *** -->
          <Item value="ToolPlaceSpec" type="QString" />
        </TOOLTYPE_ALL>
```

```
            <!-- **** For all multitools *** -->
            <TOOLTYPE_9997>
              <!-- **** Display for all multitool location types *** -->
              <Item value="MultiToolPlaceSpec" type="QString" />
            </TOOLTYPE_9997>
          </COLUMN8>
        </INSERT_COLUMN>
      </SlTmToolmagazinForm>
</LISTCONFIGURATION>
```

## Example 2: Configuring the magazine list - "Oversized tool" column is deleted

**Note**

The columns of the list configuration cannot be taken from the display in the user interface in all cases because some columns are automatically inserted and therefore not included in the count of the configuration.

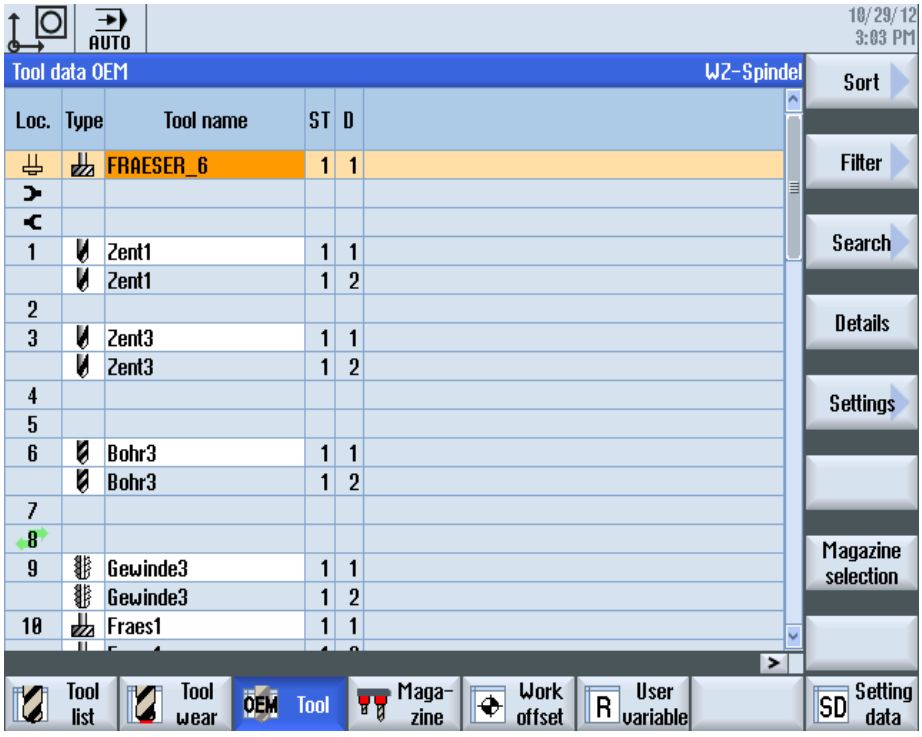
```
<LISTCONFIGURATION>
      <!-- **** Magazine list *** -->
      <SlTmToolmagazinForm>
        <!-- **** Delete column 7 *** -->
        <DELETE_COLUMN>
          <COLUMN7 />
        </DELETE_COLUMN>
      </SlTmToolmagazinForm>
</LISTCONFIGURATION>
```

## Example 3: Enabling the OEM tool data list

```
<SlTmTooloemForm>
      <Enabled value="true" type="bool" />
</SlTmTooloemForm>
```

## Result

- The "OEM Tool Data" window is displayed. It automatically displays the columns shown in the figure.
- The new "OEM tools" softkey appears and can be operated.




### 9.2.2.1 Tool parameter identifiers

## Tool parameters

For the tool parameters, the following identifiers are used in the configuration file:

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| Empty | Empty field | - |
| ToolNo | Number of the tool | - |
| AdaptNo | Adapter number assignment | $TC_MPP8 |
| DuploNo | Duplo number | $TC_TP1 |
| DuploNoRW | Duplo number, read and write access | $TC_TP1 |
| NumCuttEdges | Number of cutting edges of tool | $P_TOOLND |
| ToolIdent | Tool name | $TC_TP2 |
| ToolIdentRO | Tool name, read only access | $TC_TP2 |
| ToolInfo | Tool information | $TC_TP11 |
| ToolInMag | Magazine in which the tool is located | $A_TOOLMN |
| ToolInPlace | Magazine location containing the tool | $A_TOOLMLN |

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| ToolInMagInPlace | Magazine number/Magazine location | $A_TOOLMN / $A_TOOL_MLN |
| ToolMonTCW | Type of tool monitoring, tool life, workpiece count and wear. If wear monitoring has not been enabled via the machine data, this parameter is handled like ToolMonTC. | $TC_TP9 |
| ToolMonTC | Type of tool monitoring, tool life and workpiece count | $TC_TP9 |
| ToolPlaceSpec | Location type | $TC_TP7 |
| ToolPlaceSpecIdent | Location type, if it is to be displayed as text. The precondition is that the text for the location type was configured. | $TC_TP7 |
| ToolSearch | Tool search type for replacement tools | $TC_TP10 |
| ToolMyMag | Owner magazine of the tool | $A_MYMN |
| ToolMyPlace | Owner magazine location of the tool | $A_MYMLN |
| ToolSizeLeft | Tool size left in half locations | $TC_TP3 |
| ToolSizeRight | Tool size right in half locations | $TC_TP4 |
| ToolSizeUpper | Tool size upper in half locations | $TC_TP5 |
| ToolSizeDown | Tool size lower in half locations | $TC_TP6 |
| ToolOverSize | Tool size as fixed setting - two half locations left, two half locations right, one half location top, one half location bottom | $TC_TP3 - $TC_TP6 |
| ToolState | Tool status in the hexadecimal notation | $TC_TP8 |
| ToolStateActiv | Tool status - tool active | $TC_TP8, Bit0 |
| ToolStateEnabled | Tool status - tool released | $TC_TP8, Bit1 |
| ToolStateLocked | Tool status - tool locked | $TC_TP8, Bit2 |
| ToolStateLockedRO | Tool status - tool locked, only read access | $TC_TP8, Bit2 |
| ToolStateMeasured | Tool status - tool measured | $TC_TP8, Bit3 |
| ToolStatePrewarn | Tool status - tool has reached prewarning limit | $TC_TP8, Bit4 |
| ToolStatePrewarnRO | Tool status - tool has reached prewarning limit, only read access | $TC_TP8, Bit4 |
| ToolStateInChange | Tool status - tool is being changed | $TC_TP8, Bit5 |
| ToolStateFixed | Tool status - tool fixed place coded | $TC_TP8, Bit6 |
| ToolStateUsed | Tool status - tool was in use | $TC_TP8, Bit7 |
| ToolStateAutoReturn | Tool status - automatic return | $TC_TP8 |
| ToolStateIgnoreLocked | Tool status - do not observe locked | $TC_TP8, Bit9 |
| ToolStateMarkedToUnload | Tool status - tool is marked to unload | $TC_TP8, Bit10 |
| ToolStateMarkedToLoad | Tool status - tool is marked to load | $TC_TP8, Bit11 |
| ToolStatePermanent | Tool status - tool is permanent | $TC_TP8, Bit12 |
| ToolState1To1Exchange | Tool status - one to one replacement | $TC_TP8, Bit14 |
| ToolStateHandTool | Tool status - manual tool | $TC_TP8, Bit15 |
| ToolProtAreaFile | File name of the tool description which is required to define the protection zone | $TC_TP_PROTA |
| ToolMaxVelo | Maximum speed | $TC_TP_MAX_VELO |

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| ToolMaxAcc | Maximum acceleration | $TC_TP_MAX_ ACC |
| ToolInMultiTool | Multitool in which the tool is located | $A_TOOLMTN |
| ToolInMultiToolPlace | Multitool location containing the tool | $A_TOOLMTLN |
| ToolMyMultiTool | Owner multitool of the tool | $A_MYMTN |
| ToolMyMultiToolPlace | Owner multitool location of the tool | $A_MYMTLN |
| ToolAlarmIsExtended | PLC tool status - function "extended alarm" active | - |
| ToolAlarmLimit | PLC tool status - alarm limit reached | - |
| ToolExtAlarmLimit | PLC tool status - limit "extended alarm" reached | - |
| ToolUser_1, ... ToolUser_10 | OEM tool parameters 1 to 10 | $TC_TPC1, ... $TC_TPC10 |
| ToolAppl_1, ... ToolAppl_10 | Siemens application tool parameters 1 to 10 | $TC_TPCS1, ... $TC_TPCS10 |

*) Parameter corresponds to the system variable or the system variable is the basis for this parameter.

**Note**

**Tool name "ToolIdent"**

The maximum length of tool names is 31 ASCII characters.

The number of characters is reduced for Asian characters or Unicode characters.

The following special characters are not permitted: | # "

### 9.2.2.2 Cutting parameter identifiers

For the cutting parameters, the following identifiers are used in the configuration file:

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| EdgeNo | D number | - |
| EdgeNoIdx | Cutting edge number | - |
| ToolType | Tool type, cutting parameter 1 | $TC_DP1 |
| CuttEdgePos | Cutting edge position, cutting parameter 2 | $TC_DP2 |
| GeoLength1 | Length 1, cutting parameter 3 | $TC_DP3 |
| GeoLengthGeoAx1 | Length of geometry axis 1, cutting parameter 3, e.g. length X | $TC_DP3 |
| GeoLength | Length, cutting parameter 3 | $TC_DP3 |
| GeoLength2 | Length 2, cutting parameter 4 | $TC_DP4 |
| GeoLengthGeoAx3 | Length of geometry axis 3, cutting parameter 4, e.g. length Z | $TC_DP4 |
| GeoLength3 | Length 3, cutting parameter 5 | $TC_DP5 |
| GeoLengthGeoAx2 | Length of geometry axis 2, cutting parameter 5, e.g. length Y | $TC_DP5 |
| GeoRadius | Radius, cutting parameter 6 | $TC_DP6 |
| GeoCornerRadius | Corner radius, cutting parameter 7 | $TC_DP7 |

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| GeoOutsideRadius | Outside radius, cutting parameter 7 | $TC_DP7 |
| GeoBoomLength | Cantilever length, cutting parameter 7 | $TC_DP7 |
| GeoLength4 | Length 4, cutting parameter 8 | $TC_DP8 |
| PlateLength | Cutting tip length, cutting parameter 8 | $TC_DP8 |
| GeoLength5 | Length 5, cutting parameter 9 | $TC_DP9 |
| GeoWidth | Width, cutting parameter 9 | $TC_DP9 |
| PlateWidth | Cutting tip width, cutting parameter 9 | $TC_DP9 |
| GeoPitch | Thread pitch, cutting parameter 9 | $TC_DP9 |
| BoreRadius | Hole radius, cutting parameter 9 | $TC_DP9 |
| GeoAngle1 | Angle 1, cutting parameter 10 | $TC_DP10 |
| HolderAngle | Holder angle, cutting parameter 10 | $TC_DP10 |
| CorrectionAngle | Correction angle to align the probe, cutting parameter 10 | $TC_DP10 |
| GeoAngle2 | Angle 2, cutting parameter 11 | $TC_DP11 |
| AngleConicalMillTool | Angle conical mill tools, cutting parameter 11 | $TC_DP11 |
| CuttDirection | Reference direction holder angle, cutting parameter 11 | $TC_DP11 |
| WearLength1 | Wear length 1, cutting parameter 12 | $TC_DP12 |
| WearLengthGeoAx1 | Wear length of geometry axis 1, cutting parameter 12, e.g. Δ length X | $TC_DP12 |
| WearLength | Wear length, cutting parameter 12 | $TC_DP12 |
| WearLength2 | Wear length 2, cutting parameter 13 | $TC_DP13 |
| WearLengthGeoAx3 | Wear length of geometry axis 3, cutting parameter 13, e.g. Δ length Z | $TC_DP13 |
| WearLength3 | Wear length 3, cutting parameter 14 | $TC_DP14 |
| WearLengthGeoAx2 | Wear length of geometry axis 2, cutting parameter 14, e.g. Δ length Y | $TC_DP14 |
| WearRadius | Wear radius, cutting parameter 15 | $TC_DP15 |
| WearCornerRadius | Wear corner radius, cutting parameter 16 | $TC_DP16 |
| WearLength4 | Wear length 4, cutting parameter 17 | $TC_DP17 |
| WearLength5 | Wear length 5, cutting parameter 18 | $TC_DP18 |
| WearAngle1 | Wear angle 1, cutting parameter 19 | $TC_DP19 |
| WearAngle2 | Wear angle 2, cutting parameter 20 | $TC_DP20 |
| AdaptLength1 | Adapter length 1, cutting parameter 21 | $TC_DP21 |
| AdaptLengthGeoAx1 | Adapter length of geometry axis 1, cutting parameter 21, e.g. adapter length X | $TC_DP21 |
| AdaptLength | Adapter length, cutting parameter 21 | $TC_DP21 |
| AdaptLength2 | Adapter length 2, cutting parameter 22 | $TC_DP22 |
| AdaptLengthGeoAx3 | Adapter length of geometry axis 3, cutting parameter 22, e.g. adapter length Z | $TC_DP22 |
| AdaptLength3 | Adapter length 3, cutting parameter 23 | $TC_DP23 |
| AdaptLengthGeoAx2 | Adapter length of geometry axis 2, cutting parameter 23, e.g. adapter length Y | $TC_DP23 |
| ReliefAngle | Relief angle, cutting parameter 24 | $TC_DP24 |

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| PlateAngle | Cutting tip angle, combination of cutting parameter 24 and cutting parameter 10 | $TC_DP24 and $TC_DP10 |
| NoseAngle | Nose angle, cutting parameter 24 | $TC_DP24 |
| CuttRate | Cutting rate, cutting parameter 25 | $TC_DP25 |
| SpindleDirection | For rotating tools, direction of rotation of the tool spindle. For turning tools, direction of rotation of the main spindle | $TC_DP25 bit8 and bit9 |
| Coolant1 | Coolant 1 | $TC_DP25 Bit10 |
| Coolant2 | Coolant 2 | $TC_DP25 Bit11 |
| MFunction1 | M function 1 | $TC_DP25 Bit0 |
| MFunction2 | M function 2 | $TC_DP25 Bit1 |
| MFunction3 | M function 3 | $TC_DP25 Bit2 |
| MFunction4 | M function 4 | $TC_DP25 Bit3 |
| IsoHNoDPH | ISO H number | $TC_DPH |
| OrientNo | Cutting orientation | $TC_DPV |
| OrientV1 | Cutting orientation vector 1 | $TC_DPV3 |
| OrientV2 | Cutting orientation vector 2 | $TC_DPV4 |
| OrientV3 | Cutting orientation vector 3 | $TC_DPV5 |
| OrientVGeoAx1 | Cutting orientation vector of geometry axis 1, cutting orientation parameter 3, e.g. vector X | $TC_DPVN3 |
| OrientVGeoAx3 | Cutting orientation vector of geometry axis 3, cutting orientation parameter 4, e.g. vector Z | $TC_DPVN4 |
| OrientVGeoAx2 | Cutting orientation vector of geometry axis 2, cutting orientation parameter 5, e.g. vector Y | $TC_DPVN5 |
| TeethCount | Number of teeth, cutting parameter 34 | $TC_DPNT |
| UniqueDNo | Unique D number | $TC_DPCE |
| EdgeUser_1, ..., EdgeUser_10, | OEM cutting parameters 1 to 10 | $TC_DPC1, ... $TC_DPC10 |
| EdgeAppl_1, ..., EdgeAppl_10x | Siemens application cutting parameters 1 to 10 | $TC_DPCS1, ... $TC_DPCS10 |

*) Parameter corresponds to the system variable or the system variable is the basis for this parameter.

### 9.2.2.3 Monitoring parameter identifiers

For the monitoring parameters, the following identifiers are used in the configuration file:

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| SupWarning | Warning limits, taking the current monitoring type into consideration | - |
| SupRemaining | Actual value, taking the current monitoring type into consideration | - |
| SupDesired | Setpoint, taking the current monitoring type into consideration | - |
| SupWarningTime | Tool life warning limit | $TC_MOP1 |
| SupRemainingTime | Tool life actual value | $TC_MOP2 |

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| SupWarningPieces | Prewarning limit for workpiece count | $TC_MOP3 |
| SupRemainingPieces | Actual value for workpiece count | $TC_MOP4 |
| SupDesiredTime | Setpoint for tool life | $TC_MOP11 |
| SupDesiredPieces | Setpoint for workpiece count | $TC_MOP13 |
| SupWarningWear | Prewarning limit for wear | $TC_MOP5 |
| SupRemainingWear | Actual value for wear | $TC_MOP6 |
| SupDesiredWear | Setpoint for wear | $TC_MOP15 |
| SupExtendedAlarm | PLC tool management: Extended alarm | - |
| SupWarningPiecesIncremental | PLC tool management: Prewarning limit for workpiece count | $TC_MOP13 - $TC_MOP3 |
| SupActualPieces | PLC tool management: Actual value for workpiece count | $TC_MOP13 - $TC_MOP4 |
| EdgeSupUser_1, ... EdgeSupUser_10 | OEM cutting edge monitoring parameters 1 to 10 | $TC_MOPC1, ... $TC_MOPC10 |
| EdgeSupAppl_1, ... EdgeSupAppl_10 | Siemens application cutting edge monitoring parameters 1 to 10 | $TC_MOPCS1, ... $TC_MOPCS10 |

*) Parameter corresponds to the system variable or the system variable is the basis for this parameter.

### 9.2.2.4 Grinding parameter identifiers

For grinding parameters, the following identifiers are used in the configuration file:

| Identifier/parameter | Meaning | System variable *) |
|---|---|---|
| GrindingSpindleNo | Spindle number | $TC_TPG1 |
| GrindingConnectionRule | Chaining rule | $TC_TPG2 |
| MinimalDiscRadius | Minimum grinding wheel radius | $TC_TPG3 |
| ActualDiscRadius | Actual grinding wheel radius (read access only) | - |
| MinimalDiscWidth | Minimum grinding wheel width | $TC_TPG4 |
| ActualDiscWidth | Actual grinding wheel width | $TC_TPG5 |
| MaximalDiscSpeed | Maximum grinding wheel speed | $TC_TPG6 |
| MaximalDiscPeripheralVelocity | Maximum grinding wheel peripheral velocity | $TC_TPG7 |
| AngleBevelDisc | Angle of inclined wheel | $TC_TPG8 |
| ParamForRadiusCalculation | Parameters to calculate the radius | $TC_TPG9 |
| | Directory of the dressing program | $TC_TPG_DRSPATH |
| | Dressing program name | $TC_TPG_DRSPROG |

*) Parameter corresponds to the system variable or the system variable is the basis for this parameter.

The following parameters are assigned to the cutting edges of the grinding tool:

| Identifier/parameter | Meaning |
|---|---|
| GrindingMonitoring | Grinding-specific tool monitoring |
| GrindingUseBaseLength | Taking into account the basis dimension when calculating the grinding wheel radius |

### 9.2.2.5 Magazine location parameter identifiers

For the magazine location parameters, the following identifiers are used in the configuration file:

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| MagPlaceKind | Location kind | $TC_MPP1 |
| MagPlaceType | Location type | $TC_MPP2 |
| MagPlaceTypeIdent | Location type as text | $TC_MPP2 |
| MagPlaceTNo | Tool number of the tool in this location | $TC_MPP6 |
| MagPlaceWatchNeighbour | Consider adjacent location | $TC_MPP3 |
| MagPlaceStateLocked | Magazine location locked | $TC_MPP4, Bit0 |
| MagPlaceStateEmpty | Magazine location free | $TC_MPP4, Bit1 |
| MagPlaceStateResInterMag | Reserved for tool in buffer | $TC_MPP4, Bit2 |
| MagPlaceStateResLoadTool | Reserved for tool to be loaded | $TC_MPP4, Bit3 |
| MagPlaceStateOccupiedLeft | Magazine location occupied, left half location | $TC_MPP4, Bit4 |
| MagPlaceStateOccupiedRight | Magazine location occupied, right half location | $TC_MPP4, Bit5 |
| MagPlaceStateOccupiedUpper | Magazine location occupied, upper half location | $TC_MPP4, Bit6 |
| MagPlaceStateOccupiedDown | Magazine location occupied, lower half location | $TC_MPP4, Bit7 |
| MagPlaceStateReservedLeft | Magazine location reserved, left half location | $TC_MPP4, Bit8 |
| MagPlaceStateReservedRight | Magazine location reserved, right half location | $TC_MPP4, Bit9 |
| MagPlaceStateReservedUpper | Magazine location reserved, upper half location | $TC_MPP4, Bit10 |
| MagPlaceStateReservedDown | Magazine location reserved, lower half location | $TC_MPP4, Bit11 |
| MagPlaceMagazineNo | Magazine number | - |
| MagPlaceTypeIdx | Type index | $TC_MPP5 |
| MagPlaceWearGroup | Wear group number | $TC_MPP5 |
| MagPlaceAdaptNo | Adapter number | $TC_MPP7 |
| MagPlaceNo | Magazine location number | - |
| MagNoMagPlaceNo | Magazine number/Magazine location number | - |
| MagPlaceUser_1, ... MagPlaceUser_10 | OEM magazine location parameters 1 to 10 | $TC_MPPC1, ... $TC_MPPC10 |
| MagPlaceAppl_1, ... MagPlaceAppl_10 | Siemens magazine location parameters 1 to 10 | $TC_MPPCS1, ... $TC_MPPCS10 |

*) Parameter corresponds to the system variable or the system variable is the basis for this parameter.

### 9.2.2.6 Magazine location adapter parameter identifiers

The following identifiers are used in the configuration file for the magazine location adapter parameters:

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| MagPlaceAdapterLength1 | Magazine location adapter length 1 | $TC_ADPT1 |
| MagPlaceAdapterLengthGeoAx1 | Magazine location adapter geometry axis 1 | $TC_ADPT1 |
| MagPlaceAdapterLength2 | Magazine location adapter length 2 | $TC_ADPT2 |
| MagPlaceAdapterLengthGeoAx3 | Magazine location adapter geometry axis 3 | $TC_ADPT2 |
| MagPlaceAdapterLength3 | Magazine location adapter length 3 | $TC_ADPT3 |
| MagPlaceAdapterLengthGeoAx2 | Magazine location adapter geometry axis 2 | $TC_ADPT3 |
| MagPlaceAdapterTrafoNo | Magazine location adapter transformation number | $TC_ADPT4 |

*) Parameter corresponds to the system variable or the system variable is the basis for this parameter.

### 9.2.2.7 Multitool parameter identifiers

For the multitool parameters, the following identifiers are used in the configuration file:

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| MultiToolNo | Number of the multitool | - |
| MultiToolNumberOfPlaces | Number of multitool locations | $TC_MTPN |
| MultiToolNumberOfPlacesRO | Number of multitool locations, only read access | $TC_MTPN |
| MultiToolIdent | Name of the multitool | $TC_MTP2 |
| MultiToolInMag | Magazine in which the multitool is located | $A_TOOLMN |
| MultiToolInPlace | Magazine location containing the multitool | $A_TOOLMLN |
| MultiToolInMagInPlace | Magazine number/Magazine location | $A_TOOLMN / $A_TOOLMLN |
| MultiToolMyMag | Owner magazine of the multitool | $A_MYMN |
| MultiToolMyPlace | Owner magazine location of the multitool | $A_MYMLN |
| MultiToolPlaceSpec | Location type | $TC_MTP7 |
| MultiToolPlaceSpecIdent | Location type as text | $TC_MTP7 |
| MultiToolSizeLeft | Multitool size to the left in half locations | $TC_MTP3 |
| MultiToolSizeRight | Multitool size to the right in half locations | $TC_MTP4 |
| MultiToolSizeUpper | Multitool size to the top in half locations | $TC_MTP5 |
| MultiToolSizeDown | Multitool size to the bottom in half locations | $TC_MTP6 |
| MultiToolOverSize | Multitool size as fixed setting - two half locations left, two half locations right, one half location top, one half location bottom | $TC_MTP3 - $TC_MTP6 |
| MultiToolPosition | Multitool position | $TC_MTP_POS |
| MultiToolProtAreaFile | File name of the multitool description which is required to determine the protection zone | $TC_MTP_PROTA |
| MultiToolKindOfDist | Type of distance input of the multitool locations | $TC_MTP_KD |

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| MultiToolKindOfDistRO | Type of distance input of the multitool locations, only read access | $TC_MTP_KD |
| MultiToolDistAngle | Type of distance input of the multitool locations as an angle | $TC_MTP_KD |
| MultiToolState | Multitool status in the hexadecimal notation | $TC_MTP8 |
| MultiToolStateEnabled | Multitool status - multitool released | $TC_MTP8, Bit1 |
| MultiToolStateLocked | Multitool status - multitool locked | $TC_MTP8, Bit2 |
| MultiToolStateLockedRO | Multitool status - multitool locked, only read access | $TC_MTP8, Bit2 |
| MultiToolStateInChange | Multitool status - multitool is being changed | $TC_MTP8, Bit5 |
| MultiToolStateFixed | Multitool status - multitool fixed location coded | $TC_MTP8, Bit6 |
| MultiToolStateUsed | Multitool status - multitool was in use | $TC_MTP8, Bit7 |
| MultiToolStateAutoReturn | Multitool status - automatic return | $TC_MTP8, Bit8 |
| MultiToolStateIgnoreLocked | Multitool status - do not observe locked status | $TC_MTP8, Bit9 |
| MultiToolStateMarkedToUnload | Multitool status - multitool is selected to be unloaded | $TC_MTP8, Bit10 |
| MultiToolStateMarkedToLoad | Multitool status - multitool is selected to be loaded | $TC_MTP8, Bit11 |
| MultiToolStatePermanent | Multitool status - multitool is a master tool | $TC_MTP8, Bit12 |
| MultiToolState1To1Exchange | Multitool status - one to one replacement | $TC_MTP8, Bit14 |
| MultiToolStateHandTool | Multitool status - manual tool | $TC_MTP8, Bit15 |
| MultiToolStateLockMtIfToolLock | Multitool status - multitool is locked if a tool is locked in multitool | $TC_MTP8, Bit16 |
| MultiToolUser_1, ... MultiToolUser_10 | OEM multitool parameters 1 to 10 | $TC_MTPC1, ... $TC_MTPC10 |
| MultiToolAppl_1, ... MultiToolAppl_10 | Siemens application multitool parameters 1 to 10 | $TC_MTPCS1, ... $TC_MTPCS10 |

*) Parameter corresponds to the system variable or the system variable is the basis for this parameter.

### 9.2.2.8 Multitool location parameter identifiers

For the multitool location parameters, the following identifiers are used in the configuration file:

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| MultiToolDist | Multitool location distance taking into account the actual type of distance input of the multitool | - |
| MultiToolDistLength | Multitool location distance, length | $TC_MTPPL |
| MultiToolDistAngle | Multitool location distance angle, angle | $TC_MTPPA |
| MultiToolPlaceType | Location type | $TC_MTPP2 |
| MultiToolPlaceTypeIdent | Location type as text | $TC_MTPP2 |
| MultiToolPlaceState | Multitool status in the hexadecimal notation | $TC_MTPP4 |
| MultiToolPlaceStateLocked | Multitool location locked | $TC_MTPP4, Bit0 |

| Identifier | Parameter or property | System variable *) |
|---|---|---|
| MultiToolPlaceStateEmpty | Multitool location empty | $TC_MTPP4, Bit1 |
| MultiToolPlaceTNo | Tool number of the tool in this multitool location | $TC_MTPP6 |
| MultiToolPlaceAdaptNo | Adapter number | $TC_MTPP7 |
| MultiToolPlaceNo | Multitool location number | - |
| MultiToolPlaceUser_1, … MultiToolPlaceUser_10 | OEM multitool location parameters 1 to 10 | $TC_MTPPC1, … $TC_MTPPC10 |
| MultiToolPlaceAppl_1, … MultiToolPlaceAppl_10 | Siemens application multitool location parameters 1 to 10 | $TC_MTPPCS1, … $TC_MTPPCS10 |

*) Parameter corresponds to the system variable or the system variable is the basis for this parameter.

## 9.2.3 Configure the list parameters

### <PARAMETERCONFIGURATION> tag

In the <PARAMETERCONFIGURATION> tag, you have the following options:

1. Changing list parameters.
2. Creating new list parameters based on an existing parameter.

### Changing list parameters

You can change all the parameters included in the system apart from the ISO parameters.

Tool parameter identifiers (Page 228)

Cutting parameter identifiers (Page 230)

Monitoring parameter identifiers (Page 232)

Grinding parameter identifiers (Page 233)

Magazine location parameter identifiers (Page 234)

Magazine location adapter parameter identifiers (Page 235)

Multitool parameter identifiers (Page 235)

Multitool location parameter identifiers (Page 236)

| Entries | Meaning |
|---|---|
| HeadLine | Column header. Text entered is displayed as a header.<br>Example: Text "Radius" is displayed as a header (default). |
| Tooltip | Text that is displayed in the tooltip.<br>Example: Text "Geometry radius" is displayed in the tooltip (default). |
| ShortText | Text if the parameter is displayed in the "More Data" window.<br>Example: Text "Rad." is displayed in the "More Data" window (default). |
| DetailsText | Text, if the parameter is displayed in the "Details" window.<br>Example: Text "Radius" is displayed in the "Details" window (default). |

| Entries | Meaning | |
|---|---|---|
| Width | Column width in pixels in relation to a 640 x 480 resolution.<br>See example below: The default column width is changed to 53 pixels. | |
| DisplayMode | Values that the parameter is displayed with.<br>See example below: The default setting is changed to "DoubleMode".<br>The following values can also be accepted: | |
| | AnyMode | All characters |
| | IntegerMode | Integers |
| | UnsignedIntegerMode | Unsigned integers |
| | DoubleMode | Values with decimal places<br>The number of decimal places is specified in "DecimalPlaces" |
| | UnsignedDoubleMode | Unsigned values with decimal places<br>The number of decimal places is specified in "DecimalPlaces" |
| | Length | Specified length |
| | Angle | Specified angle |
| | LinearFeedPerTime | Linear feed in mm/minute |
| | LinearFeedPerRevolution | Linear feed in mm/revolution |
| | LinearFeedPerTooth | Linear feed in mm/tooth |
| | RevolutionSpeed | Speed |
| | ConstantCuttingSpeed | Constant cutting rate |
| DecimalPlaces | Number of decimal places if "DoubleMode" or "UnsignedDoubleMode" is selected in the DisplayMode.<br>See example: Two decimal places are entered. | |
| ItemType | Field types in which a value is displayed. | |
| | TextField | Input/output field for values and texts |
| | TextFieldReadOnly | Output field for values and texts |
| | CheckBox | Input/output field for states |
| | CheckBoxReadOnly | Output field for states |
| BitMask | Bit mask for the display of a bit from a value. The bit mask is entered as integer value. Bit 0 → 1, Bit 1 → 2, Bit 2 → 4 | |
| AccessLevel | Value for the access level that is used for these parameters in the user interface.<br>Value range 1 to 7 | |
| UpperLimit | Upper input limit for these parameters in the user interface.<br>Only applies for numerically entered parameters. | |
| LowerLimit | Lower input limit for these parameters in the user interface.<br>Only applies for numerically entered parameters. | |

## Example

In the following example, the `<GeoRadius>` parameter is used.

Only enter the modified data.

```
<PARAMETERCONFIGURATION>
    <GeoRadius>
```

```
            <Width value="53" type="int" />
            <DisplayMode value="DoubleMode" type="QString" />
            <DecimalPlaces value="2" type="int" />
        </GeoRadius>
</PARAMETERCONFIGURATION>
```

### Configuring new list parameters based on an existing list parameter.

Assign a new parameter name and enter only the changed data:

| Entries | Meaning |
|---|---|
| | Assign new parameter name<br>See 1st example: Text "NewGeoRadius" |
| Base | Name of the parameter on which the new parameter is based.<br>See 1st example: "GeoRadius" parameter is used as a template. |
| | Also enter only the changed data here. All other data is taken from the existing parameter.<br>See 1st example: Column width changed to 46 pixels. Number of decimal places changed to 1. |

### 1. Example

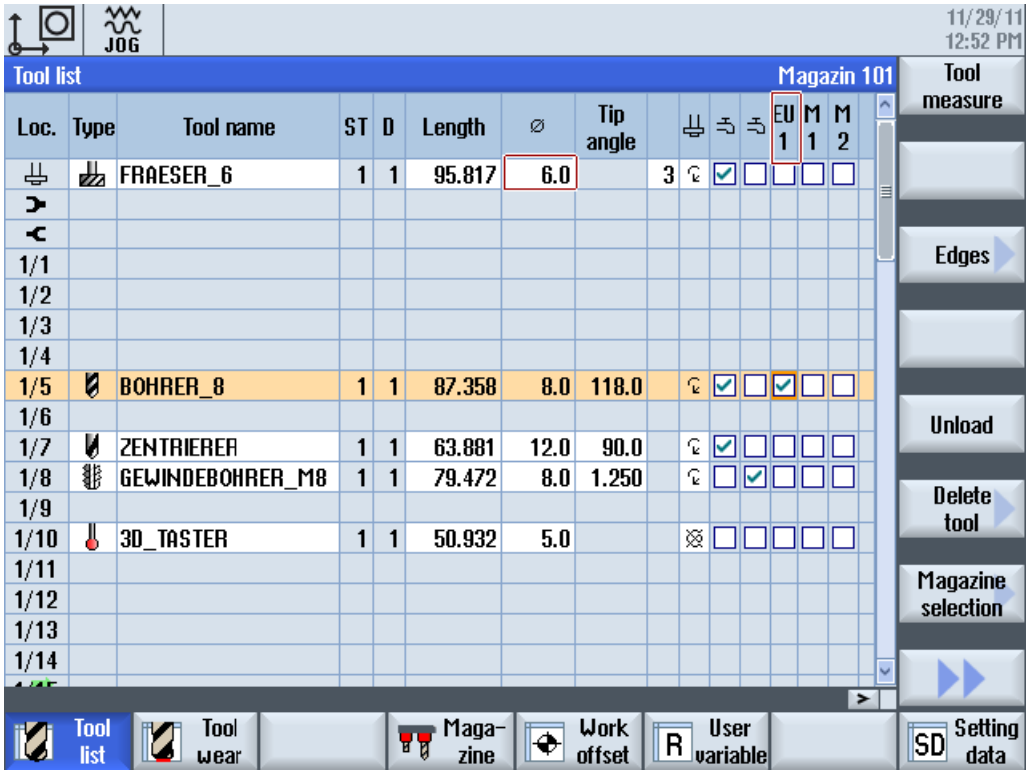
```
<PARAMETERCONFIGURATION>
    <NewGeoRadius>
        <Base value="GeoRadius" type="QString" />
        <Width value="46" type="int" />
        <DisplayMode value="DoubleMode" type="QString" />
        <DecimalPlaces value="1" type="int" />
    </NewGeoRadius>
</PARAMETERCONFIGURATION>
```

### 2. Example

| Entries | Meaning |
|---|---|
| | Assign new cutting edge parameter name<br>The name is "EdgeUser_1_Bit0" in the following example. |
| Base | Name of the parameter on which the new parameter is based.<br>In the following example, the "EdgeUser_1" parameter is used as template. |

| Entries | Meaning |
|---|---|
| | Also enter only the changed data here. All other data is taken from the existing cutting edge parameter. |
| | The following entries are changed:<br>Field type: Input/output field for states<br>Value output: All characters<br>Bit mask: Bit 0<br>Column width changed to 17 pixels.<br>Header changed to "TM_HL_EDGE_USER_1_Bit0"<br>Text in the tooltip changed to "TM_TT_EDGE_USER_1_Bit0" |

```
<PARAMETERCONFIGURATION>
    <EdgeUser_1_Bit0>
        <Base value="EdgeUser_1" type="QString" />
        <ItemType value="CheckBox" type="QString" />
        <DisplayMode value="AnyMode" type="QString" />
        <BitMask value="1" type="int" />
        <Width value="17" type="int" />
        <HeadLine value="TM_HL_EDGE_USER_1_Bit0" type="QString" />
        <ToolTip value="TM_TT_EDGE_USER_1_Bit0" type="QString" />
    </EdgeUser_1_Bit0>
</PARAMETERCONFIGURATION>
```



## 9.2.4 List of tool types

### Coding of tool types for milling tools

Group with type 1xy (milling tool):

| | |
|---|---|
| 100 | Milling tool according to CLDATA (DIN 66215) |
| 110 | Ballhead cutter (cylindrical die milling tool) |
| 111 | Ballhead cutter (tapered die milling tool) |
| 120 | End mill (without corner rounding) |
| 121 | End mill (with corner rounding) |
| 130 | Angle head cutter (without corner rounding) |
| 131 | Angle head cutter (with corner rounding) |
| 140 | Facing cutter |
| 145 | Thread cutter |
| 150 | Side mill |
| 151 | Saw |
| 155 | Bevel cutter (without corner rounding) |
| 156 | Bevel cutter (with corner rounding) |
| 157 | Tapered die-sinking cutter |
| 160 | Drill and thread milling cutter |

### Coding of tool types for drills

Group type 2xy (drills):

| | |
|---|---|
| 200 | Twist drill |
| 205 | Solid drill |
| 210 | Boring bar |
| 220 | Center drill |
| 230 | Countersink |
| 231 | Counterbore |
| 240 | Tap regular thread |
| 241 | Tap fine thread |
| 242 | Tap Whitworth thread |
| 250 | Reamer |

### Coding of tool types for grinding tools

Group type 4xy (grinding tools):

| | |
|---|---|
| 400 | Grinding wheel circumference |
| 410 | Face grinding wheel |
| 490 | Dresser |
| 494 | Dressing roll |

| | |
|---|---|
| 495 | Dressing roll with auxiliary axis |
| 496 | Dressing wheel |
| 497 | Dressing wheel with swivel axis |

Grinding tool types 401, 402, 403 as well as 411, 412, 413 are not listed as tool types that can be selected. The grinding-specific tool monitoring - taking into account the basis dimension when calculating the grinding wheel radius - is set using dedicated parameters. These parameters act on the tool types in the usual form.

### Coding of tool types for turning tools

Group type 5xy (turning tools):

| | |
|---|---|
| 500 | Roughing tool |
| 510 | Finishing tool |
| 520 | Plunge cutter |
| 530 | Parting tool |
| 540 | Threading tool |
| 550 | Steel profile |
| 560 | Rotary drill (ECOCUT) |
| 580 | Orientated probe |
| 585 | Calibration tool |

### Coding of tool types for special tools

Group type 7xy (special tools):

| | |
|---|---|
| 700 | Slotting saw |
| 710 | 3D probe |
| 711 | Edge probe |
| 712 | Mono probe |
| 713 | L probe |
| 714 | Star probe |
| 725 | Calibration tool |
| 730 | Stop |
| 731 | Spindle sleeves |
| 732 | End support |
| 900 | Auxiliary tools |

## 9.2.5 Configure tool types

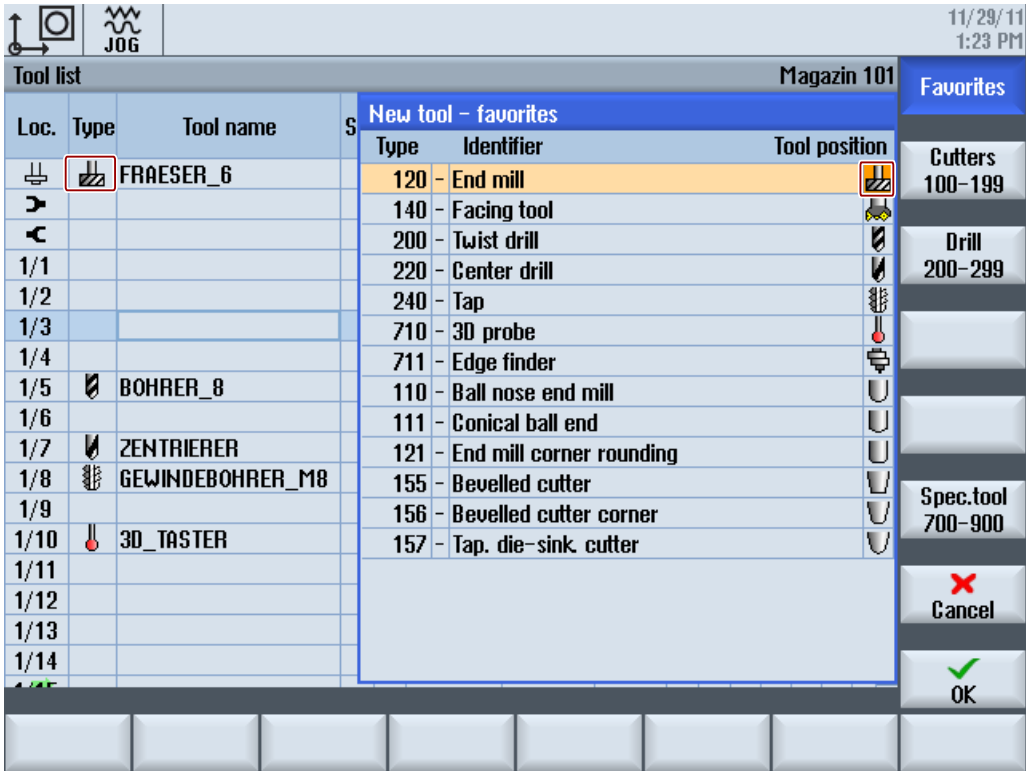
### <TOOLTYPECONFIGURATION> tag

The entries used for configuring the tool types are located in the <TOOLTYPECONFIGURATION> tag:

| Entries | Meaning |
|---|---|
| TOOLTYPE_XXX | XXX stands for the number of the tool type.<br>For the assignment of the tool type and number, please refer to Section "List of tool types (Page 241)". |
| Tooltype | Number (XXX) of the tool type |
| Name | Text identifier of the tool name. The tool name is displayed in the following windows:<br>• "New Tool - Favorites"<br>• "New Tool - Milling Cutter 100 - 199"<br>• "New Tool - Drill 200 - 299"<br>• "New Tool - Grinding Tool 400 - 499"<br>• "New Tool - Special and Auxiliary Tools 700 - 900"<br>See example: For the text identifier "TM_PAR_SHANK_END_CUTTER", the name "End mill" appears on the user interface. |
| Shortname | Text identifier of the tool name. Tool name is displayed in the tool list.<br>See example: For the text identifier "TM_PPTT_SHANK_END_CUTTER", the name "MILLING CUTTER" appears on the user interface. |
| Tooltip | Text identifier of the tool name. This tool name is displayed in the tooltip.<br>See example: For the text identifier "TM_TTTT_SHANK_END_CUTTER", the name "End mill" appears on the user interface. |
| Icon9 - for milling technology | Icon representing the tool type.<br>The icons are displayed in the following windows:<br>• "Tool List" in the "Type" column<br>• "New Tool - Favorites" in the "Tool position" column<br>• "New Tool - Milling Cutter 100 - 199" in the "Tool position" column<br>• "New Tool - Drill 200 - 299" in the "Tool position" column<br>• "New Tool - Special Tool 700 - 900 in the "Tool position" column.<br>The icons are in ".png" format. They are stored according to image resolution in the following directories:<br>/**oem**/sinumerik/hmi/ico/ico640, or ico800, or ico1024<br>/**user**/sinumerik/hmi/ico/ico640, or ico800, or ico1024 |
| IconX - for the turning and grinding technologies | Icon representing the tool type (see Icon9 - for milling technology)<br>Special feature - tool positions are supported in the turning and grinding technologies.<br>X is for the tool position represented by the icon. Position 9 is an undefined position represented by a cross. |
| Iconorder | Order in which the tool positions, represented by icons, are switched. |

### Example

```
<TOOLTYPECONFIGURATION>
    <TOOLTYPE_120>
      <Tooltype value="120" type="uint" />
      <Name value="TM_PAR_SHANK_END_CUTTER" type="QString" />
      <Shortname value="TM_PPTT_SHANK_END_CUTTER" type="QString" />
      <Tooltip value="TM_TTTT_SHANK_END_CUTTER" type="QString" />
      <Icon9 value="to_poly_shank_end_cutter_down.png" type="QString" />
    </TOOLTYPE_120>
</TOOLTYPECONFIGURATION>
```




## 9.2.6 Configuring the "More data" window

### <MOREDATACONFIGURATION> tag

The entries used for configuring the "Additional Data" window are located in the <MOREDATACONFIGURATION> tag. Various data items can be displayed for each tool type in a number of rows and columns. Each data item to be displayed is specified by entering a list parameter:

Tool parameter identifiers (Page 228)

Cutting parameter identifiers (Page 230)

Monitoring parameter identifiers (Page 232)

A brief description of the parameter then appears (ShortText), with the value itself beside it. Configure the list parameters (Page 237).

If another description text is required in the "More Data" window, a text can also be entered in a row or column. In the window, the width of the text is made to conform to that of the brief description and value of a parameter.

Any number of rows and columns can be created. A scrollbar appears when the window reaches a certain size.

| Entries | Meaning |
|---|---|
| TOOLTYPE_XXX | Number of the tool type.<br>See example: Tool type 111 = Ball end mill (tapered die-sinking cutter). |
| ROWX_COLY | Particular rowX_columnY.<br>If there is no entry after rowX and columnY, this field remains blank.<br>See example: In the "More Data" window, the first row is blank. |
| Item | The following entries are possible:<br>• Parameter identifier<br>• Text<br>See example:<br>The header "Corner radius" appears in the second row of the window.<br>The text "Radius" and an input field for the value appear in the third row of the window.<br>The fourth row is blank. |

## Example

```
<MOREDATACONFIGURATION>
    <TOOLTYPE_111>
        <ROW1_COL1>
        </ROW1_COL1>
        <ROW2_COL1>
            <Item value="TM_DGL_CORNER_RADIUS" type="QString" />
        </ROW2_COL1>
        <ROW3_COL1>
            <Item value="GeoCornerRadius" type="QString" />
        </ROW3_COL1>
        <ROW4_COL2>
        </ROW4_COL2>
    </TOOLTYPE_111>
</MOREDATACONFIGURATION>
```

## 9.2.7 Configure the "New tool - favorites" window

### <NEWTOOLFAVORITECONFIGURATION> tag

In the tag <NEWTOOLFAVORITECONFIGURATION> you define the tool types that are most frequently used. The specified tool types are displayed in the "New Tool - Favorites" window.

| Entries | Meaning |
|---|---|
| StaticTooltypes | Numbers of tool types Enter the individual numbers, separated by blanks.<br>The following section describes the assignment of the tool type and number: List of tool types (Page 241) |

### Example

```
<NEWTOOLFAVORITECONFIGURATION>
    <StaticTooltypes value="120 140 200 220 710 711" type="QString"/>
</NEWTOOLFAVORITECONFIGURATION>
```

## 9.2.8 Configuring the "New Tool" window

### <NEWTOOLCONFIGURATION> tag

You use the "New Tool" dialog to supply a new tool with specific tool data before creating the tool or before loading the tool to a magazine location. These are typically the tool size and/or the tool location type because these parameters can no longer be changed when the tool is loaded. It is also useful to display the tool type and the tool names in this window.

Specify the parameters in the <NEWTOOLCONFIGURATION> tag that are to be displayed in the "New Tool" dialog.

**Note**

The number of parameters is limited by the size of the window.

| Entries | Meaning |
|---|---|
| ItemX | Entry of the identifier of the tool parameter. "X" specifies the order in which the parameters are displayed in the window.<br>The identifier of the tool parameter can be found in Chapter:<br>Tool parameter identifiers (Page 228) |

### Example

```
<NEWTOOLCONFIGURATION>
    <!-- The standard field "oversized tool" is removed and replaced by the fields -->
```

```
    <!-- "tool location type", "tool size left" and "tool size right" -->
    <!-- -->
    <Item2 value="ToolPlaceSpec" type="QString"/>
    <Item3 value="ToolSizeLeft" type="QString"/>
    <Item4 value="ToolSizeRight" type="QString"/>
</NEWTOOLCONFIGURATION>
```

## 9.2.9 Configuring default values for new tools

### <NEWTOOLDEFAULTVALUECONFIGURATION> tag

In the <NEWTOOLDEFAULTVALUECONFIGURATION> tag, you can specify default values for tools that you create via the user interface. These values can be configured depending on the tool type and, if required, also depending on the cutting edge position:

| Entries | Meaning |
|---|---|
| TOOLTYPE_ALL | The parameter setting applies to all tool types. |
| TOOLTYPE_XXX | The parameter setting applies to a specific tool type. XXX stands for the number of the tool type. |
| CUTTEDGEPOS_XXX | The parameter setting applies to a specific cutting edge position. XXX stands for the number of the cutting edge position. |
| Default | Default value of the parameter. |

### Example

```
<NEWTOOLDEFAULTVALUECONFIGURATION>
    <!-- Initialize geometry length 1 for all tool types to 70.0 -->
    <TOOLTYPE_ALL>
        <GeoLengthGeoAx1>
            <Default value="70.0" type="double" />
        </GeoLengthGeoAx1>
    </TOOLTYPE_ALL>
    <!-- Initialize tip angle of the twist drill to 120.0 -->
    <TOOLTYPE_200>
        <NoseAngle>
            <Default value="120.0" type="double" />
        </NoseAngle>
    </TOOLTYPE_200>
    <TOOLTYPE_500>
        <!-- Initialize plate angle of the roughing tool to 82.0 -->
            <PlateAngle>
                <Default value="82.0" type="double" />
            </PlateAngle>
```

```
        <!-- Initialize holder angle of the roughing tool depending on the cutting edge position
        -->
        <HolderAngle>
               <CUTTEDGEPOS_1>
                  <Default value="93.0" type="double" />
               </CUTTEDGEPOS_1>
               <CUTTEDGEPOS_2>
                  <Default value="93.0" type="double" />
               </CUTTEDGEPOS_2>
               <CUTTEDGEPOS_3>
                  <Default value="93.0" type="double" />
               </CUTTEDGEPOS_3>
               <CUTTEDGEPOS_4>
                  <Default value="93.0" type="double" />
               </CUTTEDGEPOS_4>
               <CUTTEDGEPOS_5>
                  <Default value="49.0" type="double" />
               </CUTTEDGEPOS_5>
               <CUTTEDGEPOS_6>
                  <Default value="49.0" type="double" />
               </CUTTEDGEPOS_6>
               <CUTTEDGEPOS_7>
                  <Default value="49.0" type="double" />
               </CUTTEDGEPOS_7>
               <CUTTEDGEPOS_8>
                  <Default value="49.0" type="double" />
               </CUTTEDGEPOS_8>
        </HolderAngle>
    </TOOLTYPE_500>
    <!-- Initialize coolant 2 of the plunge cutter to switched on -->
    <TOOLTYPE_520>
        <Coolant2>
            <Default value="true" type="bool" />
        </Coolant2>
    </TOOLTYPE_520>
</NEWTOOLDEFAULTVALUECONFIGURATION>
```

## Default values when creating new tools

| System variables | | Default value |
|---|---|---|
| **Tool size** | | |
| Size on left | $TC_TP3 | 1 |
| Size on right | $TC_TP4 | 1 |
| Size above | $TC_TP5 | 1 |
| Size below | $TC_TP6 | 1 |

| System variables | | Default value |
|---|---|---|
| **Magazine location type** of the tool | | |
| Created outside of the magazine | $TC_TP7 | 1 |
| Created in a magazine location | $TC_TP7 | Location type of the magazine location |
| **Tool status** | | |
| released | $TC_TP8 | 2 |

The following combinations between the cutting-edge position and the cutting direction result dependent on the tool type. When creating the new tool, the operator selects a combination in the "New Tool - Favorites" window.

| Tool type $TC_DP1 | Default values | |
|---|---|---|
| | Cutting edge $TC_DP2 | Cutting direction $TC_DP11 |
| Type 500 roughing tool<br>Type 510 finishing tool<br>Type 560 rotary drill | 1 | 4 |
| | 2 | 3 |
| | 3 | 3 |
| | 4 | 4 |
| | 5 | 1 |
| | 6 | 3 |
| | 7 | 1 |
| | 8 | 3 |
| Type 520 plunge cutter<br>Type 530 cut-off tool<br>Grinding tool types | 1 | 2 |
| | 1 | 4 |
| | 4 | 4 |
| | 3 | 1 |
| | 4 | 1 |
| | 2 | 3 |
| | 3 | 3 |
| | 2 | 2 |
| Type 540 threading tool | 5 | 2 |
| | 5 | 1 |
| | 8 | 3 |
| | 8 | 4 |
| | 6 | 3 |
| | 6 | 4 |
| | 7 | 2 |
| | 7 | 1 |

If the ShopMill/ShopTurn option is set, the **direction of spindle rotation** is preassigned:

| Tool type $TC_DP1 | Default value $TC_DP25 | Direction of spindle rotation |
|---|---|---|
| Measuring tools | | |
| Type 580/585/710/712/713/725 | Bit 8 = 0<br>Bit 9 = 0 | Spindle stop |
| Turning tools dependent on SD54215 TM_FUNCTION_MASK_SET bit 1 | | |
| SD54215 bit 1 = 0 | Bit 8 = 1<br>Bit 9 = 0 | Spindle right |
| SD54215 bit 1 = 1 | Bit 8 = 0<br>Bit 9 = 1 | Spindle left |
| Special tools | | |
| Type 730/731/732 | Bit 8 = 0<br>Bit 9 = 0 | Spindle stop |
| All other tool types | | |
| - | Bit 8 = 1<br>Bit 9 = 0 | Spindle right |

| Tool type $TC_DP1 | Default value $TC_DP25 for switching direction |
|---|---|
| Type 712 mono probe | Bit 17 = 1<br>Bit 20 = 1 |
| Type 713 L probe | Bit 17 = 1<br>Bit 21 = 1 |
| Type 714 star probe | Bit 16 = 1<br>Bit 17 = 1<br>Bit 18 = 1<br>Bit 19 = 1 |

| Tool type $TC_DP1 | Default value $TC_DP24 for tip angle |
|---|---|
| Type 200 twist drill | 118 |
| Type 220 centering tool | 90 |

The plate angle rather than the clearance angle is displayed on the screen:
plate angle = 180 - holder angle - clearance angle

| Tool type<br>$TC_DP1 | Default values | | |
|---|---|---|---|
| | Cutting edge position<br>$TC_DP2 | Holder angle<br>$TC_DP10 | Clearance angle<br>$TC_DP24 |
| Type 500 roughing tool | 1 to 4 | 95 | 5 |
| | 5 to 8 | 50 | 50 |
| Type 510 finishing tool | 1 to 4 | 93 | 32 |
| | 5 to 8 | 62.5 | 62.5 |
| Type 560 rotary drill | - | 90 | 2 |

| Tool type $TC_DP1 | Unit of measurement | Default value $TC_DP8 for plate length |
|---|---|---|
| Type 500 roughing tool | metric | 11 |
| Type 510 finishing tool | inch | 0.4 |

For all grinding tools, the parameter for **calculating the radius**:
$TC_TPG9 = 3 (corresponds to length 1) applies

## 9.2.10 Configuring the "Details" window

### <DETAILSCONFIGURATION> tag

In the tag <DETAILSCONFIGURATION> you can make the settings at the "Detail" window:

| Entries | Meaning |
|---|---|
| ShowToolNumber | The tool number is displayed in the upper right field of the "Detail" window. |
| UseAxisNameInLength | The tool lengths are displayed with axis identifiers, e.g. length X instead of length 1.<br>If no data is specified, then the setting is taken from the tool lists. |
| ShowBaseLength | The basis length is displayed in the "Detail" window (AdaptLength1, AdaptLength2 and AdaptLength3).<br>If no data is specified, then the setting is taken from the tool lists. |
| ShowYAxis | The Y axis is displayed in the "Detail" window.<br>If no data is specified, then the setting is taken from the tool lists. |

### Example

```
<DETAILSCONFIGURATION>
    <ShowToolNumber value="true" type="bool" />
</DETAILSCONFIGURATION>
```

## 9.2.11 Assignment of TO units and toolholders to channels

### Preassignment of TO units and toolholders to channels

The assignment between TO unit and channel is made via the MD28085 machine data. The assignment between toolholder and channel is made via the MD20124 machine data (only for TE822).

Each channel is assigned its own dedicated TO unit.

- Channel 1

| MD28085 $MC_MM_LINK_TOA_UNIT | Assigns TO unit to a channel (SRAM) |
|---|---|
| = 1 | The channel is assigned TO unit 1. |

- Channel 2

| MD28085 $MC_MM_LINK_TOA_UNIT | Assigns TO unit to a channel (SRAM) |
|---|---|
| = 2 | The channel is assigned TO unit 2. |

Each channel is assigned a toolholder.

- Channel 1

| MD20124 $MC_TOOL_MANAGEMENT_TOOLHOLDER | Toolholder number |
|---|---|
| = 1 | The channel is assigned toolholder 1. |

- Channel 2

| MD20124 $MC_TOOL_MANAGEMENT_TOOLHOLDER | Toolholder number |
|---|---|
| = 1 | The channel is assigned toolholder 1. |

A revolver magazine with 12 locations, each one loading location and one toolholder, is assigned in both channels. Tasks can be uniquely assigned to the tool management through the assignment to one dedicated TO unit.

| Channel 1, TOA 1 | Channel 2, TOA 2 |
|---|---|
| Magazine 1<br>12 locations: 1/1 to 1/12 | Magazine 1<br>12 locations: 1/1 to 1/12 |
| Buffer storage magazine 9998<br>1 magazine location 9998/1 | Buffer storage magazine 9998<br>1 magazine location 9998/1 |
| Loading magazine 9999<br>1 magazine location 9999/1 | Loading magazine 9999<br>1 magazine location 9999/1 |

## Alternative configuration with TOA 1 for channel 1 and channel 2

As an alternative to pre-assignment, a common TO unit can be used for both channels.

Example with two magazines:

- Each magazine is assigned a toolholder.
- For channel 1, toolholder 1 is defined as the master toolholder.
- For channel 2, toolholder 2 is defined as the master toolholder.
- The NC program can be programmed without address extension.
- Both magazines can be loaded via the same loading location.
- The tool and magazine lists in the offset area of the HMI are the same in both channels.
- Both magazines can be loaded and positioned from both channels.

- Both channels are assigned TO unit 1.
  Channel 1:

| MD28085 $MC_MM_LINK_TOA_UNIT | Assigns TO unit to a channel (SRAM) |
|---|---|
| = 1 | The channel is assigned TO unit 1. |

  Channel 2:

| MD28085 $MC_MM_LINK_TOA_UNIT | Assigns TO unit to a channel (SRAM) |
|---|---|
| = 2 | The channel is assigned TO unit 1. |

- Each channel is assigned a toolholder.
  Channel 1:

| MD20124 $MC_TOOL_MANAGEMENT_TOOLHOLDER | Toolholder number |
|---|---|
| = 1 | The channel is assigned toolholder 1. |

  Channel 2:

| MD20124 $MC_TOOL_MANAGEMENT_TOOLHOLDER | Toolholder number |
|---|---|
| = 2 | The channel is assigned toolholder 2. |

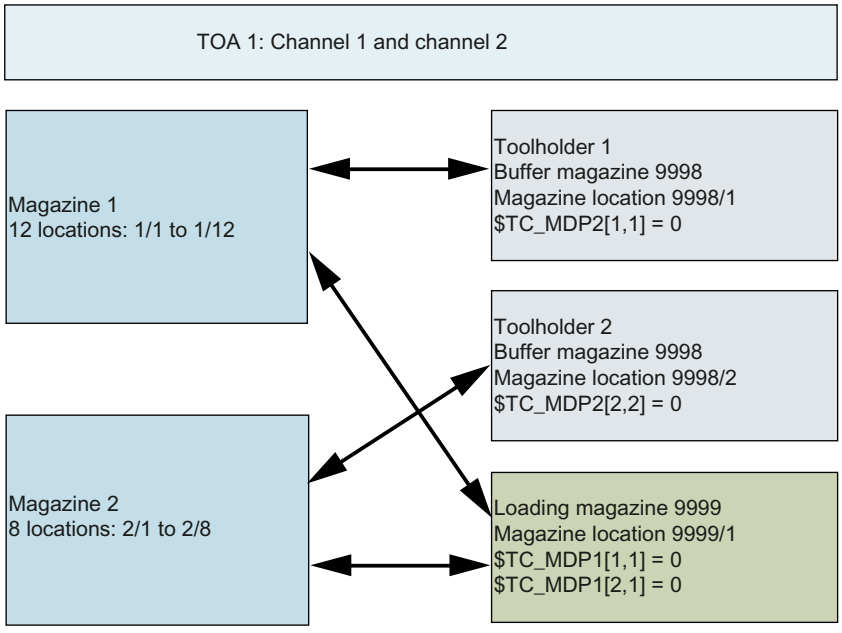



Figure 9-1 Magazine configuration

Definition of the buffer storage with allocation to the magazines:

```
$TC_MAP1[9998]=7
$TC_MAP2[9998]="Buffer1"
$TC_MAP3[9998]=17
$TC_MAP6[9998]=1
$TC_MAP7[9998]=2
$TC_MAP8[9998]=0

$TC_MPP1[9998,1]=2
$TC_MPP2[9998,1]=0
$TC_MPP3[9998,1]=0
$TC_MPP4[9998,1]=2
$TC_MPP5[9998,2]=1

$TC_MPP1[9998,2]=2
$TC_MPP2[9998,2]=0
$TC_MPP3[9998,2]=0
$TC_MPP4[9998,2]=2
$TC_MPP5[9998,2]=2
```

```
$TC_MAP1[9999]=9
$TC_MAP2[9999]="LOADINGMAGAZINE1"
$TC_MAP3[9999]=17
$TC_MAP7[9999]=1
$TC_MAP8[9999]=0

$TC_MPP1[9999,1]=7
$TC_MPP2[9999,1]=0
$TC_MPP3[9999,1]=0
$TC_MPP4[9999,1]=2

$TC_MDP1[1,1]=0
$TC_MDP1[2,1]=0

$TC_MDP2[1,1]=0
$TC_MDP2[2,2]=0
```

## 9.2.12 Assigning a name for magazines and magazine locations

### <MAGAZINEPLACENAMECONFIGURATION> tag

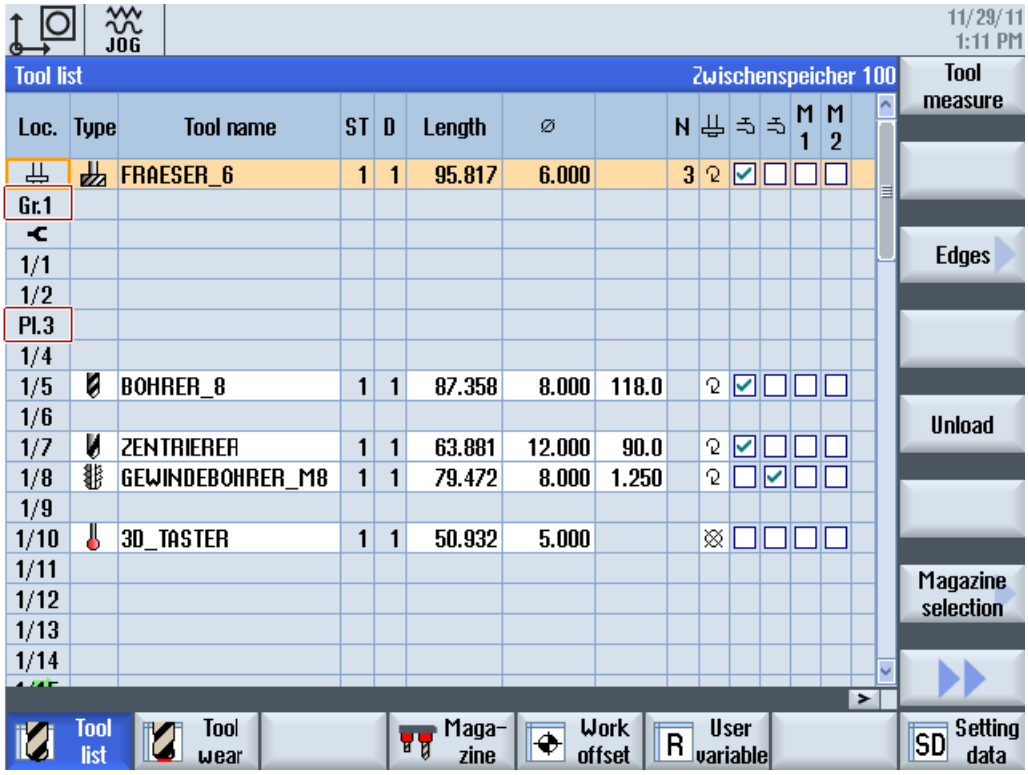
The entries to assign dedicated magazine location names are located in the <MAGAZINEPLACENAMECONFIGURATION> tag. Magazine locations can be displayed with texts. Instead of the identifier, e.g. 1/5 for the first magazine, location five, the configured text is then output in the tool lists.

| Entries | Meaning |
|---|---|
| MAGAZINE_XXX | XXX stands for the magazine number. |
| PLACE_XXX | XXX stands for the location number. |
| Name | Text identifier of the magazine location. |

### Example

```
<MAGAZINEPLACENAMECONFIGURATION>
    <MAGAZINE_9998>
        <PLACE_2>
            <Name value="TM_SPECIAL_MAG_PLACE_1" type="QString" />
        </PLACE_2>
    </MAGAZINE_9998>
    <MAGAZINE_1>
        <PLACE_3>
            <Name value="TM_SPECIAL_MAG_PLACE_2" type="QString" />
        </PLACE_3>
```

```
    </MAGAZINE_1>
</MAGAZINEPLACENAMECONFIGURATION>
```




## Name for a magazine

Dedicated texts can also be allocated for magazines. The text or the text identifier is entered into the magazine configuration file (ini file) of the NC. The system variable for the magazine name is $TC_MAP2[magNo]. The magazine name is displayed at the top right in the tool lists.

## Example

Magazine 1 should have the name "main magazine".

```
$TC_MAP2[1]="main magazine"
```

If the "main magazine" text is to be output language-dependent, then the language-dependent texts for the text identifier "main magazine" must be created.

## See also

Please refer to the following section for the procedure to create language-dependent text:

Creating OEM texts (Page 268)

## 9.2.13 Assigning a name for magazine location types

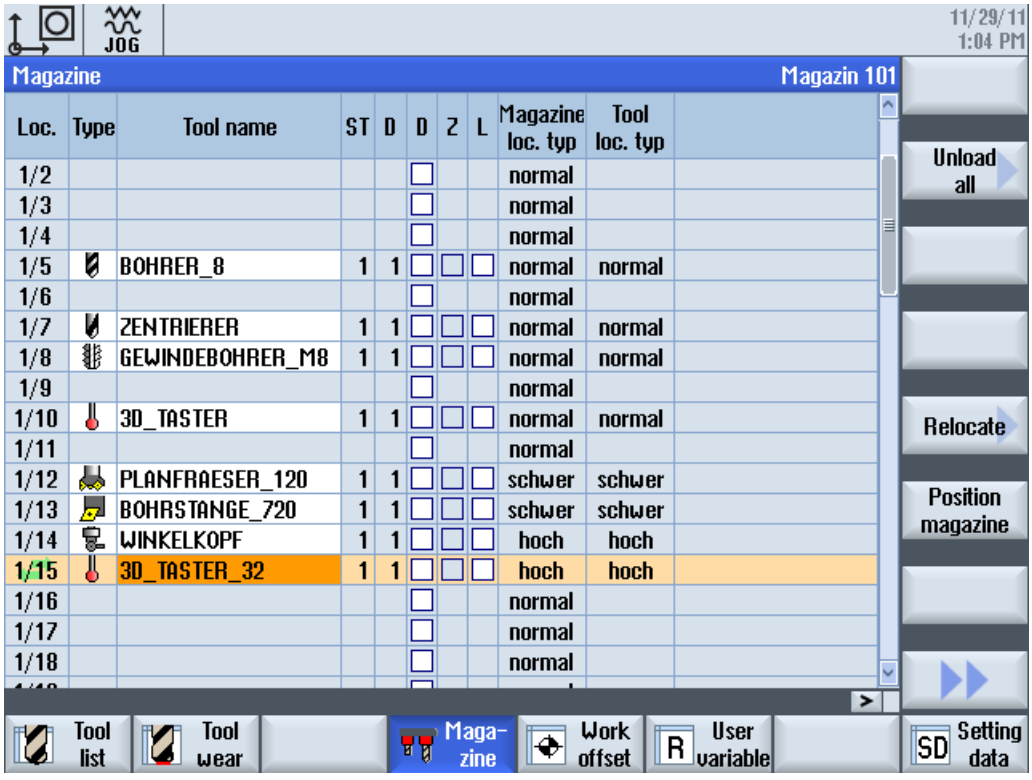
### <PLACETYPECONFIGURATION> tag

You can find entries to allocate your own location type identifiers in the tag <PLACETYPECONFIGURATION>. Magazines and magazine location types can be displayed with texts in the tool lists. Instead of identifier 2, e.g. for location type 2, the configured text is then output in the tool lists.

| Entries | Meaning |
|---|---|
| PLACETYPEXXX | XXX stands for the number of the location type |
| Text | Text identifier of the location type |
| Tooltip | Tooltip text identifier of the location type |

### Example

```
<PLACETYPECONFIGURATION>
    <PLACETYPE1>
       <Text value="TM_PLACETYPE_1" type="QString" />
       <Tooltip value="TM_TT_PLACETYPE_1" type="QString" />
    </PLACETYPE1>
    <PLACETYPE2>
       <Text value="TM_PLACETYPE_2" type="QString" />
       <Tooltip value="TM_TT_PLACETYPE_2" type="QString" />
    </PLACETYPE2>
    <PLACETYPE3>
       <Text value="TM_PLACETYPE_3" type="QString" />
       <Tooltip value="TM_TT_PLACETYPE_3" type="QString" />
    </PLACETYPE3>
</PLACETYPECONFIGURATION>
```




## See also

Please refer to the following section for the procedure to create language-dependent text:

Creating OEM texts (Page 268)

## 9.2.14 Assignment of magazines to channels

### Requirement

Turning technology has been set: Name of the associated configuration file sltmturninglistconfig.xml.

### Assignment between magazine and channel

With this setting, magazines can be assigned to channels. This makes sense if the machine is configured so that several channels are assigned to one tool area (TOA area). If each magazine is then only used in one channel, here it is possible to specify this assignment between the magazine and channel.

As a consequence of this configuration, the tool icons are output with reference to their position depending on the channel. Every channel can be assigned a position using the channel machine data MD52000 MCS_DISP_COORDINATE_SYSTEM. If a tool is at a magazine

location which is assigned a channel via this configuration, then the tool icon is output depending on this position that has been set.

This setting option is only available for turning. This can be especially helpful if, on a multi-channel machine, a magazine is used before the turning center and another magazine after the turning center. This means that the tools can be entered into the tool list in a position visible for the operator.

### <CHANNELMAGAZINEASSIGNMENT> tag

Entries to assign magazines to channels are in the <CHANNELMAGAZINEASSIGNMENT> tag.

| Entries | Meaning |
|---|---|
| CHANNELNO_XXX | XXX stands for the channel number. |
| MAGAZINES | Number of the magazines that are assigned to the channel. |

### Example

The following assignment between magazine and channel should apply:

| | |
|---|---|
| Magazine 1-3 | Channel 1 |
| Magazine 4 | Channel 2 |
| Magazine 5 | Channel 3 |
| Magazine 6 | Channel 4 |

```
<CHANNELMAGAZINEASSIGNMENT>
   <CHANNELNO_1>
      <Magazines value="1 2 3" type="QString" />
   </CHANNELNO_1>
   <CHANNELNO_2>
      <Magazines value="4" type="QString" />
   </CHANNELNO_2>
   <CHANNELNO_3>
      <Magazines value="5" type="QString" />
   </CHANNELNO_3>
   <CHANNELNO_4>
      <Magazines value="6" type="QString" />
   </CHANNELNO_4>
</CHANNELMAGAZINEASSIGNMENT>
```

## 9.2.15 Coolant and tool-specific functions

### Assigning coolant (ShopMill/ShopTurn)

In the tool management of ShopMill/ShopTurn, you can assign coolant and tool-specific functions to each tool. These are activated when the tool is changed. You assign the coolant to the corresponding machine functions (M functions) using the following machine data:

| | |
|---|---|
| MD52230 $MCS_M_CODE_ALL_COOLANTS_OFF | M code for all coolants OFF |
| MD52231 $MCS_M_CODE_COOLANT_1_ON | M code for coolant 1 ON |
| MD52232 $MCS_M_CODE_COOLANT_2_ON | M code for coolant 2 ON |
| MD52233 $MCS_M_CODE_COOLANT_1_AND_2_ON | M code for both coolants ON |

### Tool-specific functions (ShopMill)

Tool-specific functions 1 ... 4 are intended for other M functions that can be activated for a tool, for example, third coolant, speed monitoring, tool breakage, etc.

Up to eight M functions can be programmed for the activation or deactivation of the tool-specific functions. Definition is realized using the following machine data:

| | |
|---|---|
| MD52281 $MCS_TOOL_MCODE_FUNC_ON[0...3] | M code for tool-specific function ON |
| = - 1 | |

| | |
|---|---|
| MD52282 $MCS_TOOL_MCODE_FUNC_OFF[0...3] | M code for tool-specific function OFF |
| = - 1 | |

The M functions for the tool-specific functions are executed by the ShopMill cycles at the tool change. If the checkbox is activated, the corresponding M code is generated from MD52281. If the checkbox is not activated, the corresponding M code is generated from MD52282.

If you do not want to automatically execute additional M commands at the tool change, then set the associated machine data to -1 for the M commands of this tool-specific function. This also hides the associated fields in the tool list.

The default value of the machine data is -1 so that no M values are output for the corresponding functions.

### Technical procedure

The M functions for the tool-specific functions are generated by the ShopMill cycles in a set (up to four M commands are possible).

The number and also the sequence of the M commands are dependent on the settings of the machine data (MD52281 to MD52282) and the programming.

If M functions less than 100 are used, the number and sequence of the M commands has no significance. They are located in the user PLC in the decoded area (DB250x.DBB1000 to DB250x.DBB1012), independent of the number and the sequence when programming.

However, if M functions from 100 and higher are used, then you must decode these yourself in the user PLC. In this case, you must take into consideration the number and sequence (M function 1 to M function 4).

### JOG mode

In the JOG mode, no coolant and tool-specific functions are output to the interface. For a tool change in the JOG mode, these functions can be initiated by the operator using the keys on the machine control panel (implementation via the PLC user program).

### Example: Machine data settings

| | |
|---|---|
| MD52281 $MCS_TOOL_MCODE_FUNC_ON[0] | = 90 |
| MD52281 $MCS_TOOL_MCODE_FUNC_ON[1] | = 92 |
| MD52281 $MCS_TOOL_MCODE_FUNC_ON[2] | = 94 |
| MD52281 $MCS_TOOL_MCODE_FUNC_ON[3] | = 96 |

| | |
|---|---|
| MD52282 $MCS_TOOL_MCODE_FUNC_OFF[0] | = 91 |
| MD52282 $MCS_TOOL_MCODE_FUNC_OFF[1] | = - 1 |
| MD52282 $MCS_TOOL_MCODE_FUNC_OFF[2] | = 95 |
| MD52282 $MCS_TOOL_MCODE_FUNC_OFF[3] | = 97 |

As a result, fields 1 to 4 of the tool-specific functions are displayed in the tool list.

The following fields are activated:

| | |
|---|---|
| Tool-specific function 1: | On |
| Tool-specific function 2: | Off |
| Tool-specific function 3: | Off |
| Tool-specific function 4: | Off |

The M functions M90, M95 and M97 are generated by the ShopMill cycles at the tool change.

You can change the texts for "Coolant 1/2 on/off" and "Tool-spec. fct. 1, ..." (see section Identifiers of the standard texts (Page 269)). So that the texts between the tool list and program area machine functions match, please note the following section.

### Creating user-specific texts for the "Machine Functions" window.

If you require an additional possibility of switching on or switching off coolant and tool-specific functions, then you can program this in the "Program" operating area using the input window "Machine Functions". You can create the texts and the tooltips for the tool-specific functions which are used in this window.

Define the texts for the M functions in the file "slstepforms_xxx.ts".

If you require the texts in additional languages, then you must create a separate file for each language. The only difference in the files is that the file name has the language code "xxx". Use only the specified language codes.

## File "slstepforms_xxx.ts "

| Tag | Meaning |
|---|---|
| source | Text identifier for label: SlStepLabels<br>A maximum of up to four M functions can be defined:<br>T_LAB_USER_MACHINE_FUNC_1<br>T_LAB_USER_MACHINE_FUNC_2<br>T_LAB_USER_MACHINE_FUNC_3<br>T_LAB_USER_MACHINE_FUNC_4 |
| | Text identifier for tooltip: SlStepToolTip<br>There are 3 tooltips for every machine function x (1 - 4):<br>T_TT_USER_MACHINE_FUNC_x: Standard tooltip if the machine function was not selected.<br>T_TT_USER_MACHINE_FUNC_x _ON: Tooltip when the machine function is activated<br>T_TT_USER_MACHINE_FUNC_x_OFF: Tooltip for deactivated machine function |
| translation | Text, which is displayed at the user interface.<br>Label texts: You can only enter texts on one line. The "lines" entry is not applicable.<br>Tooltip: You can enter multi-line texts. A line break is set using the "%n" character. |
| chars | Text length<br>The maximum text length is 30 characters. |
| lines | Number of lines<br>The maximum number of lines is three. |
| remark | Own comment - entry is not displayed at the user interface. |

## Procedure

1. You can copy the sample file "oem_slstepforms_deu.ts" from the following directory: /**siemens**/sinumerik/hmi/template/lng
2. Save or create the file in the /**oem**/sinumerik/hmi/lng or /**user**/sinumerik/hmi/lng directory.
3. Change the file name to "slstepforms_deu.ts".
   If you require texts for additional languages, then a separate file must be created for each language. Save the file with the appropriate language code in the file name.
4. Open the file and define the texts in the **<message>** and **</message>** areas.
5. Restart the system.
   The file must still be converted into a binary format in order that the texts are displayed during the program runtime. This data is only converted during power up.

## Example of a file "slstepforms_deu.ts"

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE TS><TS>
<context>
   <name>SlStepLabels</name>
   <message>
```

```
      <source>T_LAB_USER_MACHINE_FUNC_1</source>
      <translation>tool-specific function 1</translation>
      <chars>20</chars>
   </message>
<context>
   <name>SlStepToolTip</name>
   <!-- tooltips of user machine function 1-->
   <message>
      <source>T_TT_USER_MACHINE_FUNC_1</source>
      <translation>User%nMachine functions 1</translation>
      <chars>25</chars>
      <lines>3</lines>
   </message>
</context
</TS>
```

## 9.2.16 Reason for the tool change when reactivating

### Precondition



**Software option**

You require the "MC Information System TDI Statistic" option for this function

### Function

When reactivating a tool, you have the option of specifying the reason for the tool change, e.g. to use this information for statistical purposes. The "Reason for Tool Change" selection window is opened after pressing the "Reactivate" softkey. In a checkbox, you can select the reason for the tool change. The tool change is entered in a tool parameter to be configured or in a local variable of the SlCap service.

### Tool change reasons

You can adapt or expand the tool change reasons as required. The operating software provides the following pre-defined texts for the configuration:

| Text identifier | Text |
|---|---|
| TM_DGL_DESIRED_PIECES_ELAPSED | Unit quantity setpoint (desired pieces) elapsed |
| TM_DGL_TOOL_BREAKAGE | Tool breakage |
| TM_DGL_CUTTING_EDGE_DAMAGE | Cutting edge damage |
| TM_DGL_PREMATURE_WEAR | Premature wear |
| TM_DGL_TYPE_REEQUIPPING | Type reequipping |
| TM_DGL_TOOL_TRIAL | Tool trial |
| TM_DGL_PREVENTIVE_CHANGE | Preventive change |

### <TOOLCHANGEREASON> tag

You configure the function in the <TOOLCHANGEREASON> tag:

| Parameter | Designation |
|---|---|
| Enabled | true - "Reason for tool change" selection window is displayed when re-activating. |
| | false - default value, selection window is not displayed. |
| OnlyOneReasonAllowed | It is only permissible to select one tool change reason. |
| UseCapLocalParameter | The reason for the tool change is entered into a local variable of the SlCap service. |
| REASONXXX | XXX stands for an internal number. The numbers for the tool change reason must be clear and unique. |
| Text | Text identifier of the tool change reason. |
| Parameter | Parameter identifier of the parameter in which it is entered whether the tool change reason was selected. |
| BitMask | Bit mask to enter the reason for the change in the local variable of the SlCap service. The bit mask is entered as integer value. Bit 0 → 1, bit 1 → 2, bit 2 → 4 |

### Local variable of the SICap service

The identifier of the local variables is called "/Hmi/ToolEdgeReplacementReason". The variable is the string type.

The table below lists the most important arguments and their significance:

| Argument | Meaning |
|---|---|
| datetime | Data and time of the event |
| containerNo | Number of the management area |
| toaNo | Number of the To area |
| toolNo | Tool number |
| edgeNo | Cutting edge number |
| toolIdent | Tool name |
| toolDuplo | Replacement tool number |
| ncuName | Name of the NCU |
| reactivationReason | Reason for tool change |
| monitoringMode | Monitoring type |
| remainingBefore | Remaining unit quantity/tool lifetime before reactivation |

### Example 1

The following example uses the texts for the tool change reason, already prepared in the operating software. In this example, the tool change reason is stored in parameters that still have to be configured, e.g. the OEM tool data 1, bit 1 and following.

```
<TOOLCHANGEREASON>
   <Enabled value="true" type="bool" />
```

```
    <REASON1>
      <Text value="TM_DGL_DESIRED_PIECES_ELAPSED" type="QString" />
      <Parameter value="ToolUser_1_Bit1" type="QString" />
    </REASON1>
    <REASON2>
      <Text value="TM_DGL_TOOL_BREAKAGE" type="QString" />
      <Parameter value="ToolUser_1_Bit2" type="QString" />
    </REASON2>
    <REASON3>
      <Text value="TM_DGL_CUTTING_EDGE_DAMAGE" type="QString" />
      <Parameter value="ToolUser_1_Bit3" type="QString" />
    </REASON3>
    <REASON4>
      <Text value="TM_DGL_PREMATURE_WEAR" type="QString" />
      <Parameter value="ToolUser_1_Bit4" type="QString" />
    </REASON4>
    <REASON5>
      <Text value="TM_DGL_TYPE_REEQUIPPING" type="QString" />
      <Parameter value="ToolUser_1_Bit5" type="QString" />
    </REASON5>
    <REASON6>
      <Text value="TM_DGL_TOOL_TRIAL" type="QString" />
      <Parameter value="ToolUser_1_Bit6" type="QString" />
    </REASON6>
    <REASON7>
      <Text value="TM_DGL_PREVENTIVE_CHANGE" type="QString" />
      <Parameter value="ToolUser_1_Bit7" type="QString" />
    </REASON7>
</TOOLCHANGEREASON>
```

## Example 2

Also in this example, the prepared texts for the reason for the change are used. It is only permitted to enter one reason for the change as a valid entry. The reason for the tool change is saved in the local variable of the SICap service.

```
<TOOLCHANGEREASON>
   <Enabled value="true" type="bool" />
    <OnlyOneReasonAllowed value="true" type="bool" />
    <UseCapLocalParameter value="true" type="bool" />
    <REASON1>
      <Text value="TM_DGL_DESIRED_PIECES_ELAPSED" type="QString" />
      <BitMask value="1" type="int" />
    </REASON1>
```

```
    <REASON2>
      <Text value="TM_DGL_TOOL_BREAKAGE" type="QString" />
      <BitMask value="2" type="int" />
    </REASON2>
    <REASON3>
      <Text value="TM_DGL_CUTTING_EDGE_DAMAGE" type="QString" />
      <BitMask value="4" type="int" />
    </REASON3>
    <REASON4>
      <Text value="TM_DGL_PREMATURE_WEAR" type="QString" />
      <BitMask value="8" type="int" />
    </REASON4>
    <REASON5>
      <Text value="TM_DGL_TYPE_REEQUIPPING" type="QString" />
      <BitMask value="16" type="int" />
    </REASON5>
    <REASON6>
      <Text value="TM_DGL_TOOL_TRIAL" type="QString" />
      <BitMask value="32" type="int" />
    </REASON6>
    <REASON7>
      <Text value="TM_DGL_PREVENTIVE_CHANGE" type="QString" />
      <BitMask value="64" type="int" />
    </REASON7>
</TOOLCHANGEREASON>
```

### See also

For configuring new parameters, see Configure the list parameters (Page 237).

## 9.2.17 Configure the code carrier connection

### Requirement



**Software option**

You require the "Access MyTool ID" option for this function (Tool Ident Connection).

## <IDENTCONNECTIONCONFIGURATION> tag

The entries used for configuring the code carrier connection in the tool list of the user interface are located in the <IDENTCONNECTIONCONFIGURATION> tag. The code carrier connection is implemented via TDI Ident Connection.

The following functions in the tool list are available:

1. Creating tools for the code carrier:

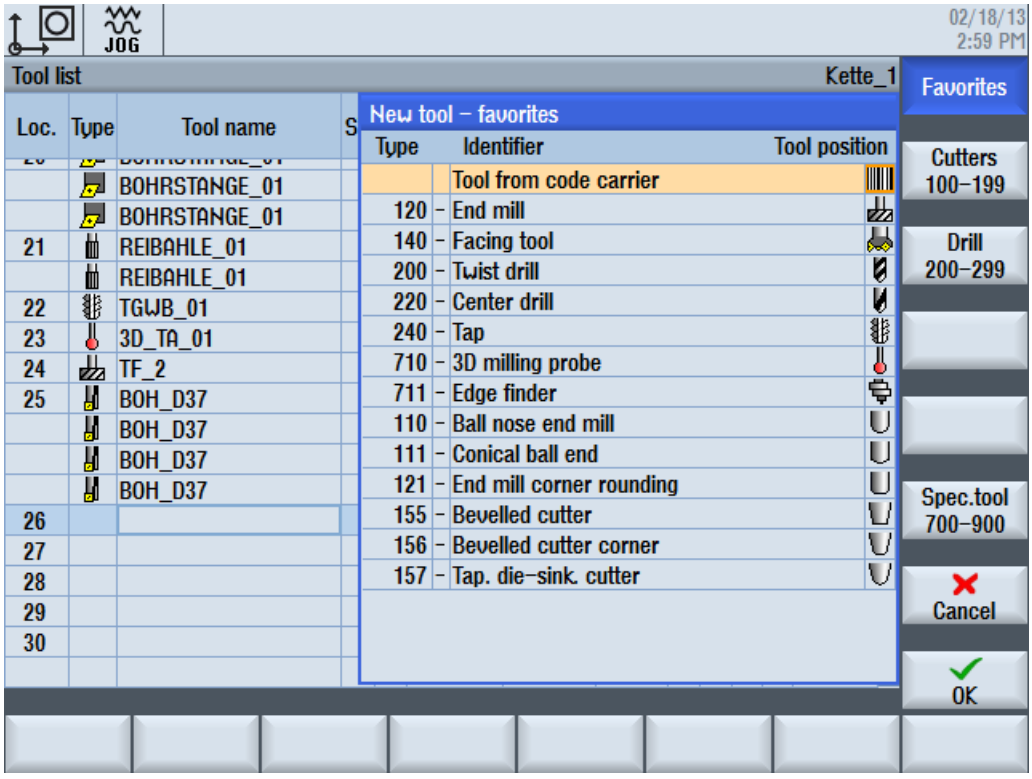



Figure 9-2 New tool for the code carrier

The data for the code carrier tool is read and displayed in the "New Tool" window. Click OK to create the tool in the NC and to initialize it with the code carrier data.

2. Unloading tools on the code carrier:
   The tool is unloaded and the data of the tool is written to the code carrier. The tool can also be deleted using a setting (see below).
3. Deleting the tool on the code carrier:
   The tool is unloaded and the data of the tool is written to the code carrier. The tool is then deleted from the NC. Using a setting (see below), the "On code carrier" softkey can also be hidden for deleting a tool.

The following settings can be made:

| Entries | Meaning |
|---|---|
| Enabled | true - code carrier connection is offered.<br>false - code carrier connection is not offered. |
| ShowInternalMessage | true - internal messages for error analysis are also displayed.<br>false - default setting, no internal messages are displayed. |
| UnloadWithDelete | true - the tool is deleted when unloaded to the code carrier. The "On code carrier" function is not offered for "Delete tool".<br>false - default setting, the tool is only unloaded for "unload to code carrier" and remains in the NC memory. |
| OnlyInLoadStation | true - default setting, on the one hand, only loaded tools are stored "on code carrier", on the other hand, new tools "from code carrier" can be created only directly in the magazine.<br>false - tools can be created independently of the "from code carrier" magazine slot or deleted "on code carrier". |
| CreateNewToolWithDialog | true - default setting, before creating the tool for code carrier, the tool data is displayed in the "New Tool" window.<br>false - the tool is created directly in the tool list. |

## Example

```
<IDENTCONNECTIONCONFIGURATION>
      <Enabled value="true" type="bool" />
      <ShowInternalMessage value="false" type="bool" />
      <UnloadWithDelete value="false" type="bool" />
      <OnlyInLoadStation value="true" type="bool" />
      <CreateNewToolWithDialog value="true" type="bool" />
</IDENTCONNECTIONCONFIGURATION>
```

## References

For further information about tool management, please refer to:

Function Manual, SINUMERIK Integrate Access MyTool Ident Connection (MCIS TDI Ident Connection)

# 9.3 Creating OEM texts

## 9.3.1 Creating OEM texts

### Overview

This section describes how you create OEM texts in a particular language.

The texts are created in the "sltmlistdialog_xxx.ts" file.

A dedicated file must be created for each language. The file names differ by the language code. For "xxx", set the appropriate language code.

### Creating texts

All OEM texts from the previous sections, such as the tooltip, can be made language-dependent by using the OEM text as text identifier under the "source" tag. The translation of the text in the particular language is entered under the "translation" tag.

| Entries | Meaning |
|---|---|
| source | Text identifier<br>See 1st example: Single-line text with new text identifier "MY_NEW_TEXT".<br>See 2nd example: Double-line text with new text identifier "MY_NEW_TEXT_2_LINES". |
| translation | Text, which is displayed on the user interface.<br>You can enter single and multi-line texts. "%n" is used to indicate a line break.<br>See 1st example: Single-line text "My new text" is displayed.<br>See 2nd example: Double-line text "My new - text" is displayed. |
| chars | Text length<br>See 1st example: Text length of 30 characters is specified.<br>See 2nd example: Text length of 10 characters per line is specified. |
| lines | Number of lines<br>See 2nd example: The number of rows is 2. |
| remark | Own comments - are not displayed. |

### Procedure

1. You can copy the sample file "oem_sltmlistdialog_eng.ts" from the following directory: /**siemens**/sinumerik/hmi/template/lng
2. Save the file in the /**oem**/sinumerik/hmi/lng or /**user**/sinumerik/hmi/lng directory.
3. Change the file name to "sltmlistdialog_eng.ts".
   If you wish to create texts for additional languages, then a separate file must be created for each language. Save the file with the appropriate language code in the file name.

4. Open the file and define the texts in the <message> and </message> areas.
5. Restart the system.
   The file must still be converted into a binary format in order that the texts are displayed during the program runtime. This data is only converted during power up.

## 9.3.2 Identifiers of the standard texts

### Identifiers of the standard texts

With the help of the identifiers, you can change the standard texts of OEM parameters, coolants and tool-specific functions. This applies to the column header in the tool lists, the tooltip and, if available, also the designation in the detail displays.

A list of the identifiers is provided below:

### Identifiers of OEM tool parameters

"ToolUser_1" to "ToolUser_10"

"ToolAppl_1" to "ToolAppl_10"

| Column heading | Tooltip |
|---|---|
| TM_HL_TOOL_USER_1 to ...10 | TM_TT_TOOL_USER_1 to ...10 |
| TM_HL_TOOL_APPL_1 to ...10 | TM_TT_TOOL_APPL_1 to ...10 |

| Tool details | Tool details - all parameters |
|---|---|
| TM_HL_TOOL_USER_1_DETAILS to ...10_DETAILS | TM_HL_TOOL_USER_1_LONG to ...10_LONG |
| TM_HL_TOOL_APPL_1_DETAILS to ...10_DETAILS | TM_HL_TOOL_APPL_1_LONG to ...10_LONG |

### Identifiers of the OEM cutting edge parameters

"EdgeUser_1" to "EdgeUser_10"

"EdgeAppl_1" to "EdgeAppl_10"

| Column heading | Tooltip |
|---|---|
| TM_HL_EDGE_USER_1 to ...10 | TM_TT_EDGE_USER_1 to ...10 |
| TM_HL_EDGE_APPL_1 to ...10 | TM_TT_EDGE_APPL_1 to ...10 |

| Tool details | Tool details - all parameters |
|---|---|
| TM_HL_EDGE_USER_1_DETAILS to ...10_DETAILS | TM_HL_EDGE_USER_1_LONG to ...10_LONG |
| TM_HL_EDGE_APPL_1_DETAILS to ...10_DETAILS | TM_HL_EDGE_APPL_1_LONG to ...10_LONG |

### Identifiers of the OEM monitoring parameters

"EdgeSupUser_1" to "EdgeSupUser_10"

"EdgeSupAppl_1" to "EdgeSupAppl_10"

| Column heading | Tooltip |
|---|---|
| TM_HL_EDGE_SUPUSER_1 to ...10 | TM_TT_EDGE_SUPUSER_1 to ...10 |
| TM_HL_EDGE_SUPAPPL_1 to ...10 | TM_TT_EDGE_SUPAPPL_1 to ...10 |

| Tool details | Tool details - all parameters |
|---|---|
| TM_HL_EDGE_SUPUSER_1_DETAILS to ...10_DETAILS | TM_HL_EDGE_SUPUSER_1_LONG to ...10_LONG |
| TM_HL_EDGE_SUPAPPL_1_DETAILS to ...10_DETAILS | TM_HL_EDGE_SUPAPPL_1_LONG to ...10_LONG |

### Identifiers of OEM multitool parameters

"MultiToolUser_1" to "MultiToolUser_10"

"MultiToolAppl_1" to "MultiToolAppl_10"

| Column heading | Tooltip |
|---|---|
| TM_HL_MULTITOOL_USER_1 to ...10 | TM_TT_MULTITOOL_USER_1 to ...10 |
| TM_HL_MULTITOOL_APPL_1 to ...10 | TM_TT_MULTITOOL_APPL_1 to ...10 |

| Tool details | Tool details - all parameters |
|---|---|
| TM_HL_MULTITOOL_USER_1_DETAILS to ... 10_DETAILS | TM_HL_MULTITOOL_USER_1_LONG to ...10_LONG |
| TM_HL_MULTITOOL_APPL_1_DETAILS to ... 10_DETAILS | TM_HL_MULTITOOL_APPL_1_LONG to ...10_LONG |

### Identifiers of the OEM magazine location parameters

"MagPlaceUser_1" to "MagPlaceUser_10"

"MagPlaceAppl_1" to "MagPlaceAppl_10"

| Column heading | Tooltip |
|---|---|
| TM_HL_MAGPLACE_USER_1 to ...10 | TM_TT_MAGPLACE_USER_1 to ...10 |
| TM_HL_MAGPLACE_APPL_1 to ...10 | TM_TT_MAGPLACE_APPL_1 to ...10 |

### Identifiers of OEM multitool location parameters

"MultiToolPlaceUser_1" to "MultiToolPlaceUser_10"

"MultiToolPlaceAppl_1" to "MultiToolPlaceAppl_10"

| Column heading | Tooltip |
|---|---|
| TM_HL_MTPLACE_USER_1 to ...10 | TM_TT_MTPLACE_USER_1 to ...10 |
| TM_HL_MTPLACE_APPL_1 to ...10 | TM_TT_MTPLACE_APPL_1 to ...10 |

### Identifiers of the coolant and tool-specific functions

"Coolant1" and "Coolant2"

"MFunction1" to "MFunction4"

| Column heading | Tooltip |
|---|---|
| No column | TM_TT_STATE_COOL_1 and ...2 |
| TM_HL_MFCT1 to ...4 | TM_TT_MFCT1 to ...4 |

## 9.3.3 Examples of OEM texts

### Example 1 with a single-line and double-line text

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE TS><TS>
<!--      ********************************************************** -->
<!--      Definition of OEM texts
<!--      ********************************************************** -->
<context>
   <name>SlTmListForm</name>
<!--      ********************************************************** -->
<!--      enter your text behind this comment -->
<!--      ********************************************************** -->
<!--      ********************************************************** -->
<!--      1st example of a single-line text -->
<!--      ********************************************************** -->
<!--   <message>
         <source>MY_NEW_TEXT</source>
         <translation>My new text</translation>
         <chars>30</chars>
       </message>-->
<!--      ********************************************************** -->
<!--      2nd example of a double-spaced text -->
<!--      ********************************************************** -->
<!--   <message>
         <source>MY_NEW_TEXT_2_LINES</source>
         <translation>My new%n text</translation>
         <remark>My comment to the text</remark>
         <chars>10</chars>
```

```
        <lines>2</lines>
      </message>-->
<!--      ********************************************************** -->
<!--      enter your text ahead of this comment -->
<!--      ********************************************************** -->
</context>
</TS>
```

## Example 2

A new parameter is configured in the following example from Section Configure the list parameters (Page 237):

```
<PARAMETERCONFIGURATION>
    <EdgeUser_1_Bit0>
        <Base value="EdgeUser_1" type="QString" />
        <ItemType value="CheckBox" type="QString" />
        <DisplayMode value="AnyMode" type="QString" />
        <BitMask value="1" type="int" />
        <Width value="17" type="int" />
        <HeadLine value="TM_HL_EDGE_USER_1_Bit0" type="QString" />
        <ToolTip value="TM_TT_EDGE_USER_1_Bit0" type="QString" />
    </EdgeUser_1_Bit0>
</PARAMETERCONFIGURATION>
```

The two texts under HeadLine and ToolTip should be output depending on the language. To do this, the two text identifiers must be included in the sltmlistdialog_xxx.ts file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE TS><TS>
<!--      ********************************************************** -->
<!--      Definition of OEM texts
<!--      ********************************************************** -->
<context>
   <name>SlTmListForm</name>
<!--      ********************************************************** -->
<!--      enter your text behind this comment -->
<!--      ********************************************************** -->
<!--      ********************************************************** -->
<!--      Translate the text for the headline -->
<!--      ********************************************************** -->
      <message>
        <source>TM_HL_EDGE_USER_1_Bit0</source>
        <translation>heavy %n tool</translation>
        <chars>10</chars>
        <lines>2</lines>
      </message>
```

```
<!--    ********************************************************** -->
<!--    Translate the text for the tooltip -->
<!--    ********************************************************** -->
      <message>
        <source>TM_TT_EDGE_USER_1_Bit0</source>
        <translation>heavy tool</translation>
        <chars>20</chars>
      </message>
<!--    ********************************************************** -->
<!--    enter your text ahead of this comment -->
<!--    ********************************************************** -->
</context>
</TS>
```

# 9.4 Examples

## 9.4.1 Example: Configuring an OEM tool list

### 9.4.1.1 Adapting the configuration file

The following excerpt shows the contents of the configuration file:

- Milling technology: sltmlistconfig.xml
- Turning technology: sltmturninglistconfig.xml

The file in this example is stored in the "user/sinumerik/hmi/cfg" directory.

This example describes a configuration of the OEM tool list.

- OEM tool parameter 1 is displayed in column 6.
- A new "Coolant 3" parameter is inserted in column 7 that displays bit 0 from OEM tool parameter 2.
- OEM cutting edge parameter 1 is displayed in column 8.

#### Example

"sltmlistconfig.xml" or "sltmturninglistconfig.xml" file:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CONFIGURATION>
   <!-- **** Identifier for the configuration of the lists *** -->
   <LISTCONFIGURATION>
     <!-- **** OEM tool list *** -->
```

```
  <SlTmTooloemForm>
    <!-- **** Enabling the OEM tool list *** -->
    <Enabled value="true" type="bool" />
    <!-- **** Columns 1 to 5 have already been assigned default values. *** -->
    <!-- **** Column 6 *** -->
    <COLUMN6>
      <!-- **** For all tool types *** -->
      <TOOLTYPE_ALL>
        <!-- **** Display OEM tool parameter 1 *** -->
        <Item value="ToolUser_1" type="QString" />
      </TOOLTYPE_ALL>
    </COLUMN6>
    <!-- **** Column 7 *** -->
    <COLUMN7>
      <!-- **** For all tool types *** -->
      <TOOLTYPE_ALL>
        <!-- **** Priority tool, own parameter *** -->
        <!-- **** Display OEM tool parameter 2 bit 0 *** -->
        <Item value="PriorTool" type="QString" />
      </TOOLTYPE_ALL>
    </COLUMN7>
    <!-- **** Column 8 *** -->
    <COLUMN8>
      <!-- **** For all tool types *** -->
      <TOOLTYPE_ALL>
        <!-- **** Display OEM cutting edge parameter 1 *** -->
        <Item value="EdgeUser_1" type="QString" />
      </TOOLTYPE_ALL>
    </COLUMN8>
  </SlTmTooloemForm>
</LISTCONFIGURATION>
<!-- **** Identifier for the configuration of own parameter *** -->
<PARAMETERCONFIGURATION>
  <!-- **** Change OEM tool parameter 1 *** -->
  <ToolUser_1>
    <!-- **** Column width *** -->
    <Width value="36" type="int" />
    <!-- **** Type change to unsigned integer *** -->
    <DisplayMode value="UnsignedIntegerMode" type="QString" />
  </ToolUser_1>
  <!-- **** Identifier of own parameter *** -->
  <!-- **** Priority tool *** -->
  <PriorTool>
    <!-- ** New parameter is based on OEM tool parameter 2**-->
```

```
        <Base value="ToolUser_2" type="QString" />
        <!-- **** Bit 0 is evaluated *** -->
        <BitMask value="1" type="int" />
        <!-- **** Text identifier of the headline *** -->
        <HeadLine value="TMO_HL_PRIOR_TOOL" type="QString" />
        <!-- **** Text identifier of the tooltip *** -->
        <ToolTip value="TMO_TT_PRIOR_TOOL" type="QString" />
        <!-- **** Column width *** -->
        <!-- **** Recommended widths 640x480 - 19 pixels *** -->
        <!-- **** 800x600 - 21, 1024x768 - 17 *** -->
        <Width value="19" type="int" />
        <!-- **** Display item of the parameter *** -->
        <ItemType value="CheckBox" type="QString" />
      </PriorTool>
    </PARAMETERCONFIGURATION>
</CONFIGURATION>
```

### 9.4.1.2 Adaptation of the customer text file

The following excerpt shows the contents of the "sltmlistdialog_eng.ts" customer text file.

The file in this example is stored in the "user/sinumerik/hmi/lng" directory.

## Example

"sltmlistdialog_eng.ts" file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE TS><TS>

<!-- ************************************************************ -->
<!-- Definition of OEM texts                                      -->
<!-- ************************************************************ -->
<context>
     <name>SlTmListForm</name>

<!-- ************************************************************ -->
<!-- Enter your text behind this comment                          -->
<!-- ************************************************************ -->

        <!-- **** OEM tool parameter 1 headline *** -->
        <!--  **** Weight  *** -->
        <message>
          <!-- **** Text identifier *** -->
          <source>TM_HL_TOOL_USER_1</source>
```

```
  <!--  **** Text that is output *** -->
  <translation>%nweight</translation>
  <!--  **** Comment only for the translator *** -->
  <remark>priority tool</remark>
  <!--  **** Maximum line length in characters *** -->
  <!--  **** only for the translator *** -->
  <chars>7</chars>
  <!--  **** Number of possible lines *** -->
  <!--  **** only for the translator *** -->
  <lines>2</lines>
</message>
<!-- **** OEM tool parameter 1 tooltip *** -->
<!--  **** Weight  *** -->
<message>
  <source>TM_TT_TOOL_USER_1</source>
  <translation>weight of the tool</translation>
  <chars>35</chars>
  <lines>2</lines>
</message>
<!--  **** Headline for own parameter *** -->
<!-- **** Priority tool *** -->
<message>
  <source>TMO_HL_PRIOR_TOOL</source>
  <translation>V</translation>
  <remark>priority tool</remark>
  <chars>1</chars>
  <lines>2</lines>
</message>
<!--  **** Tooltip for own parameter *** -->
<!-- **** Priority tool *** -->
<message>
  <source>TMO_TT_PRIOR_TOOL</source>
  <translation>priority tool, %nuse tool preferably< /translation>
  <chars>35</chars>
  <lines>3</lines>
</message>
<!--  **** OEM cutting edge parameter 1 headline *** -->
<!--  **** Maximum tool width  *** -->
<message>
  <source>TM_HL_EDGE_USER_1</source>
  <translation>maximum%nwidth</translation>
  <chars>8</chars>
  <lines>2</lines>
</message>
```

```
        <!--  **** OEM cutting edge parameter 1 tooltip *** -->
        <!--  **** Maximum tool width  *** -->
        <message>
          <source>TM_TT_EDGE_USER_1</source>
          <translation>maximum width%nof the tool%noverall</translation>
          <chars>35</chars>
          <lines>2</lines>
        </message>

<!--  ************************************************************ -->
<!--  Enter your text ahead of this comment                        -->
<!--  ************************************************************ -->

</context>
</TS>
```

## 9.4.2 Example: Configuring magazine location types with names

### 9.4.2.1 Adapting the configuration file

The following excerpt shows the contents of the configuration file:

- Milling technology: sltmlistconfig.xml
- Turning technology: sltmturninglistconfig.xml

The file in this example is stored in the "user/sinumerik/hmi/cfg" directory.

This example describes a configuration of the location types in the magazine list.

In this example, the magazine location type and the tool location type are to be displayed as a toggle field with plain texts.

The assignment of the parameters to the columns of the list is made in the "LISTCONFIGURATION" area. The magazine location type is displayed in column 10. The tool location type is displayed in column 11.

The texts to the location types are assigned in the "PLACETYPECONFIGURATION" area. The number of the location type is the number at the end of the "PLACETYPE" identifier. A text identifier for a toggle box entry and a text identifier for a tooltip can be stored for each location type.

### Example

"sltmlistconfig.xml" or "sltmturninglistconfig.xml" file:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CONFIGURATION>
    <!-- **** Configuration of the lists *** -->
```

```
<LISTCONFIGURATION>
  <!-- **** Magazine list *** -->
  <SlTmToolmagazinForm>
    <!-- **** Column 10 *** -->
    <COLUMN10>
      <!-- **** For all tool types *** -->
      <TOOLTYPE_ALL>
        <!-- **** Display magazine location type *** -->
        <Item value="MagPlaceTypeIdent" type="QString" />
      </TOOLTYPE_ALL>
    </COLUMN10>
    <!-- **** Column 11 *** -->
    <COLUMN11>
      <!-- **** For all tool types *** -->
      <TOOLTYPE_ALL>
        <!-- **** Display tool location type *** -->
        <Item value="ToolPlaceSpecIdent" type="QString" />
      </TOOLTYPE_ALL>
      <!-- **** For all multitools *** -->
      <TOOLTYPE_9997>
        <!-- **** Display for all multitool location types *** -->
        <Item value="MultiToolPlaceSpecIdent" type="QString" />
      </TOOLTYPE_9997>
    </COLUMN11>
  </SlTmToolmagazinForm>
</LISTCONFIGURATION>
<!-- **** Configuration of the magazine location types *** -->
<PLACETYPECONFIGURATION>
  <!-- **** Magazine location type 1 *** -->
  <PLACETYPE1>
    <!-- **** Text identifier of the toggle box entry *** -->
    <Text value="TMMP_ITEM_NORMAL_TOOL" type="QString" />
    <!-- **** Text identifier of the tooltip *** -->
    <Tooltip value="TMMP_TT_NORMAL_TOOL" type="QString" />
  </PLACETYPE1>
  <!-- **** Magazine location type 2 *** -->
  <PLACETYPE2>
    <Text value="TMMP_ITEM_HEAVY_TOOL" type="QString" />
    <Tooltip value="TMMP_TT_HEAVY_TOOL" type="QString" />
  </PLACETYPE2>
  <!-- **** Magazine location type 3 *** -->
  <PLACETYPE3>
    <Text value="TMMP_ITEM_BIG_TOOL" type="QString" />
    <Tooltip value="TMMP_TT_BIG_TOOL" type="QString" />
```

```
    </PLACETYPE3>
    <!-- **** Magazine location type 4 *** -->
    <PLACETYPE4>
      <Text value="TMMP_ITEM_SENSITIVE_TOOL" type="QString" />
      <Tooltip value="TMMP_TT_SENSITIVE_TOOL" type="QString" />
    </PLACETYPE4>
  </PLACETYPECONFIGURATION>
</CONFIGURATION>
```

### 9.4.2.2 Adaptation of the customer text file

The following excerpt shows the contents of the "sltmlistdialog_eng.ts" customer text file.

The file in this example is stored in the "user/sinumerik/hmi/lng" directory.

## Example

"sltmlistdialog_eng.ts" file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE TS><TS>

<!-- ************************************************************ -->
<!--  Definition of OEM texts                                     -->
<!-- ************************************************************ -->
<context>
    <name>SlTmListForm</name>

<!-- ************************************************************ -->
<!--  Enter your text behind this comment                         -->
<!-- ************************************************************ -->
    <!-- **** Magazine location type 1 item *** -->
    <!--  **** Normal tool *** -->
    <message>
      <!-- **** Text identifier *** -->
      <source>TMMP_ITEM_NORMAL_TOOL</source>
      <!--  **** Text that is output *** -->
      <translation>normal</translation>
      <!--  **** Maximum line length in characters *** -->
      <!--  **** only for the translator *** -->
      <chars>9</chars>
    </message>
    <!-- **** Magazine location type 1 tooltip *** -->
    <!--  **** Normal tool *** -->
    <message>
```

```
  <source>TMMP_TT_NORMAL_TOOL</source>
  <translation>normal tool,%nno restrictions</translation>
  <chars>35</chars>
  <lines>2</lines>
</message>
<!-- **** Magazine location type 2 item *** -->
<!-- **** Heavy tool *** -->
<message>
  <source>TMMP_ITEM_HEAVY_TOOL</source>
  <translation>heavy</translation>
  <chars>9</chars>
</message>
<!-- **** Magazine location type 2 tooltip *** -->
<!-- **** Heavy tool *** -->
<message>
  <source>TMMP_TT_HEAVY_TOOL</source>
  <translation>heavy tool</translation>
  <chars>35</chars>
  <lines>2</lines>
</message>
<!-- **** Magazine location type 3 item *** -->
<!-- **** Big tool *** -->
<message>
  <source>TMMP_TT_BIG_TOOL</source>
  <translation>big</translation>
  <chars>9</chars>
</message>
<!-- **** Magazine location type 3 tooltip *** -->
<!-- **** Big tool *** -->
<message>
  <source>TMMP_TT_BIG_TOOL</source>
  <translation>big tool</translation>
  <chars>35</chars>
  <lines>2</lines>
</message>
<!-- **** Magazine location type 4 item *** -->
<!-- **** Sensitive tool *** -->
<message>
  <source>TMMP_ITEM_SENSITIVE_TOOL</source>
  <translation>sensitive</translation>
  <chars>9</chars>
</message>
<!-- **** Magazine location type 4 tooltip *** -->
<!-- **** Sensitive tool *** -->
```

```
    <message>
      <source>TMMP_TT_SENSITIVE_TOOL</source>
      <translation>Notice: Tool can%nbe destroyed easily</translation>
      <chars>35</chars>
      <lines>2</lines>
    </message>

<!-- ************************************************************ -->
<!--  Enter your text ahead of this comment                       -->
<!-- ************************************************************ -->

</context>
</TS>
```

## 9.5 Replace T, M or D function

With the following machine data and system variables you replace a T, M or D function with a cycle.

For information about tool change cycles, see Section "Manufacturer cycle for tool change CUST_T and CUST_M6 (Page 79)".

### References

- SINUMERIK 828D Commissioning Manual, CNC Commissioning; Tool Management
  "Application example for turning machine" section
  "Application example for milling machine" section
- SINUMERIK 840D sl Function Manual Tool Management

### Machine data

| MD10715 $MN_M_NO_FCT_CYCLE | Enter the M function that is to be replaced. |
|---|---|
| = 6 | Default value |

| MD10716 $MN_M_NO_FCT_CYCLE_NAME | Name of the cycle that should be executed instead of the M function. |
|---|---|
| = L6 | Default value |

| MD10717 $MN_T_NO_FCT_CYCLE_NAME | Name of the cycle that should be executed instead of the T function. |
|---|---|
| = TCHANGE | Default value |

| MD11717 $MN_D_NO_FCT_CYCLE_NAME | | Name of the cycle that should be executed instead of the D function. |
|---|---|---|
| = Compensation | Example value | |

| MD15710 $MN_TCA_CYCLE_NAME | | Name of the cycle that should be executed instead of the TCA command. |
|---|---|---|
| = TCHANGE | Example value | |

## System variables of the replacement cycles

| System variable | Meaning |
|---|---|
| $C_T | T number of the tool (numerical). |
| $C_T_PROG | Bool variable<br>This variable indicates whether a T word is available in $C_T. |
| $C_TS | This variable returns the string of the tool identifier. |
| $C_TS_PROG | Bool variable<br>This variable indicates whether an identifier is available in $C_TS |
| $C_TE | This variable returns the address extension of the T word. |
| $C_TCA | Bool variable<br>This variable indicates whether the TCA replacement is active |
| $C_DUPLO_PROG | Bool variable<br>This variable indicates whether the Duplo number has been programmed for the TCA replacement. |
| $C_DUPLO | For $C_DUPLO_PROG == TRUE, this variable contains the value of the programmed duplo number. |
| $C_THNO_PROG | Bool variable<br>This variable indicates whether the toolholder/spindle number has been programmed for the TCA replacement. |
| $C_THNO | For $C_THNO_PROG == TRUE, this variable contains the value of the programmed toolholder/spindle number |
| $C_D | This variable contains the programmed D number. |
| $C_D_PROG | Bool variable<br>This variable indicates whether an offset number is available in $C_D |
| $C_DL | This variable contains the programmed D total/setup offset. |
| $C_DL_PROG | Bool variable<br>This variable indicates whether an offset number is available in $C_DL. |
| $C_ME | This variable contains the address extension of the M word. |
| $C_M | This variable contains the value of the replaced address M (integer). |
| $C_M_PROG | This variable contains the value TRUE, if the M function was programmed for the tool change. |

## System variables of the tool management interface

| System variable | Meaning |
|---|---|
| $AC_TC_FCT | Function number (command number of the NC) |
| $AC_TC_STATUS | Status of the command to be read via $AC_TC_FCT. |
| $AC_TC_THNO | Number of the toolholder (spindle number) where the new tool should be loaded. |
| $AC_TC_TNO | Internal T number of the tool to be replaced or prepared. |
| $AC_TC_MFN | Source magazine number of the new tool. |
| $AC_TC_LFN | Source location number of the new tool. |
| $AC_TC_MTN | Target magazine number of new tool. |
| $AC_TC_LTN | Target location number of the new tool. |
| $AC_TC_MFO | Source magazine number of the old tool (to be removed). |
| $AC_TC_LFO | Source location number of the old tool (to be removed). |
| $AC_TC_MTO | Source magazine number of the old tool (to be removed). |
| $A_TC_LTO | Target location number of the old tool (to be removed). |
| $AC_TC_CMDT | Trigger variable to output command of the NC (is set for one IPO). |
| $AC_TC_ACKT | Trigger variable for acknowledgment of the PLC (is set for one IPO). |
| $AC_TC_CMDC | Counter for the command output. |
| $AC_TC_ACKC | Counter for the acknowledgments. |
| $AC_TC_MMYN | Owner magazine number of the new tool (to be loaded). |
| $AC_TC_LMYN | Owner location number of the new tool (to be loaded). |

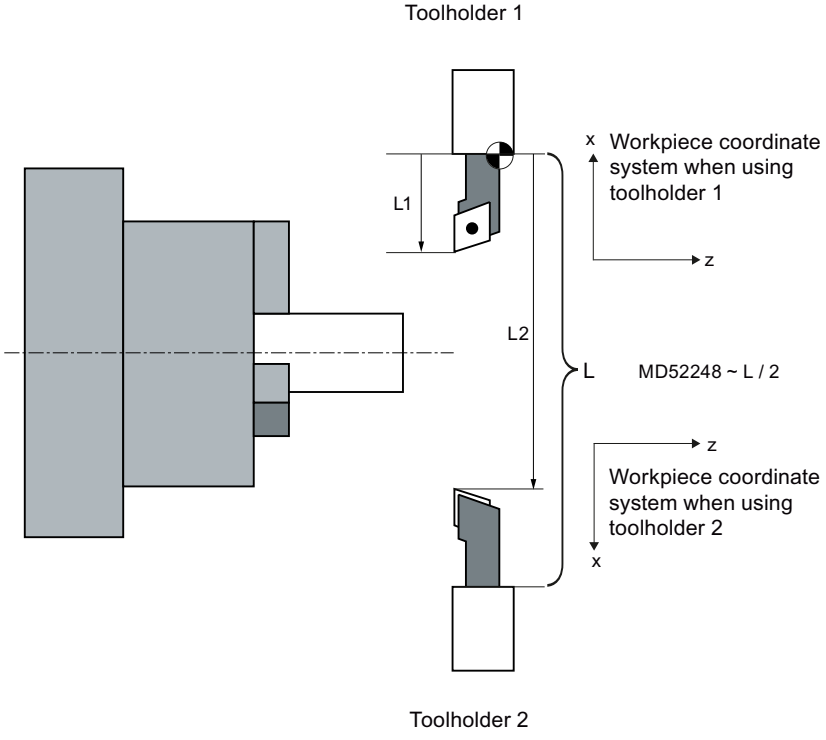
## Supplementary system variables for a multitool

| System variable | Meaning |
|---|---|
| $AC_TC_TOOLIS | Type of the tool be transported<br>0 = single tool<br>1 = multitool with location number as distance coding<br>2 = multitool with length separation as distance coding<br>3 = multitool with angular separation as distance coding |
| $AC_TC_MTDIST | Distance of the new tool in the multitool from the reference location. If the multitool has the length separation as distance coding, the variable contains the length in the unit mm or inch as specified by the set dimension system. |
| $AC_TC_MTNLOC | Number of locations of the new multitool present in the tool change or tool transport. |
| $AC_TC_MTTN | Number of the multitool in which the new tool is to be deployed. |
| $AC_TC_MTLTN | Location number of the multitool with the new tool. |

# 9.6 Working with two toolholders

It is possible to work on a single turning machine with two toolholders, both of which are mounted on an X axis. The toolholders may be revolvers, multifix, or a combination of both. As both tool holders are mounted on the same axis it is only possible to machine with one tool. The workpiece is located between the two tool adapters.

All tools have the same reference point (on toolholder 1) irrespective of which toolholder they are on. Consequently, the tool lengths of the second toolholder are always longer than the tool lengths of the first toolholder. The main machining is performed in the negative X axis direction.




## Toolholder definition

The MD52248 REV_2_BORDER_TOOL_LENGTH defines a tool length limit X so that the tools can be assigned to the two toolholders automatically:

Tool length X < MD52248: Tool belongs to the 1st toolholder
Tool length X ≥ MD52248: Tool belongs to the 2nd toolholder

For the mirror X to become active for the activation of the tools of the second toolholder, the following T/TCA replacement must be set:

**MD10717 $MN_T_NO_FCT_CYCLE_NAME="F_T_REV2"**

MD15710 $MN_TCA_CYCLE_NAME="F_T_REV2"

**Note**

If there are a large number of old G code programs that were programmed without mirroring of the X axis, you should consider setting mirroring for G code programs.

If the required mirroring is also to be automatically activated when programming the TRANSMIT command, the TRANSMIT command must be replaced by a subprogram (e.g. in the manufacturer directory):

```
MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[0]="TRANSMIT"
MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[1]="TRANSM"
```

TRANSMIT.SPF subprogram:

```
PROC TRANSMIT(INT_NR)
F_T_REV2
TRANSM(_NR)
F_T_REV2
RET
```

**Note**

In the status display, TRANSM is now displayed instead of TRANSMIT.

## Additional parameter assignments

For the used mirrorings of the X and Y axes and the offsets of the C axis, the following machine data has to be configured:

| MD24006 $MC_CHSFRAME_RESET_MASK | Active system frames after RESET |
|---|---|
| Bit 4 = 1 | System frame for workpiece reference remains active after RESET |

| MD24007 $MC_CHSFRAME_RESET_CLEAR_MASK | Clear system frames after RESET |
|---|---|
| Bit 4 = 0 | Do not clear system frame workpiece reference after RESET |

| MD28082 $MC_MM_SYSTEM_FRAME_MASK | Setting-up system frames (SRAM) |
|---|---|
| Bit 4 = 1 | Workpiece reference ($P_WPFRAME) |

| MD28083 $MC_MM_SYSTEM_DATAFRAME_MASK | Setting-up data management, system frames (SRAM) |
|---|---|
| Bit 4 = 1 | Workpiece reference ($P_WPFRAME) |

Due to the coordinate system being mirrored, the following machine/setting data must be set:

| MD21202 $MC_LIFTFAST_WITH_MIRROR | Lift fast with mirror |
|---|---|
| = 1 | For the determination of the lifting direction, it is also mirrored for active mirroring. The mirroring of the lifting direction only refers to the direction components perpendicular to the tool direction. |

| SD42910 $SC_MIRROR_TOOL_LENGTH | Sign change, tool wear when mirroring |
|---|---|
| = 1 | Inverting the sign mirrors the wear values of the tool length components whose associated axes are mirrored. |

### General conditions

- The taper turning is not enabled for "working with two toolholders".
- The B-axis function (kinematics) is also not enabled for "working with two toolholders".

## 9.7 Deploy multitool

### 9.7.1 Enable the display of a multitool

#### Displaying a multitool in the tool list

You make the settings for the tool lists in the "sltmturninglistconfig.xml" (turning) file in the "User" or "OEM" directories of the control. For example, the XML file is located in the "System-CF-Card/oem/sinumerik/hmi/cfg" directory.

For a detailed description of the procedure and setting options, see Section "Tool management, Configuring the user interface (Page 217)". A template for the settings is provided in the directory "/siemens/sinumerik/hmi/template/cfg" with the name "oem_sltmlistconfig.xml".

To display the multitool, you must set the "DisableMultiTools“ entry to the "false“ vale in the configuration file.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CONFIGURATION>
    <SETTINGS>
        <DisableMultiTools value="false" type="bool" />
    </SETTINGS>
</CONFIGURATION>
```

### Example

The following example is typical for turning machines with a main spindle and a counterspindle. For multitools, two locations are permanently set with "Location" distance coding. Location 1 receives the "Toolholder for main spindle" tooltip; location 2 receives the "Toolholder for counterspindle" tooltip.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CONFIGURATION>
    <SETTINGS>
        <DisableMultiTools value="false" type="bool" />
        <FixNumberOfMultiToolPlaces value="2" type="int" />
        <FixKindOfDistInMultiTools value="1" type="int" />
    </SETTINGS>
    <MULTITOOLPLACETOOLTIPCONFIGURATION>
        <PLACE_1>
            <ToolTip value="TM_HOLDER_OF_MAIN_SPINDLE" type="QString" />
        </PLACE_1>
        <PLACE_2>
            <ToolTip value="TM_HOLDER_OF_COUNTER_SPINDLE" type="QString" />
        </PLACE_2>
    </MULTITOOLPLACETOOLTIPCONFIGURATION>
</CONFIGURATION>
```

## 9.7.2 Loading and unloading multitool

A multitool is loaded and unloaded into or out of the magazine just the same as loading or unloading an individual tool.

### Acknowledgment from the PLC and information about the multitool position

Acknowledgment is realized using position value "0" (DB40xx.DBWy004). "0" is a permitted value and means that the multitool is not positioned. If a multitool is directly loaded to the spindle or to a revolver, then after selecting the location or initializing using reset, the tool that is located at machine data MD20274 $MC_MULTITOOLLOC_DEFAULT is activated.

## Example

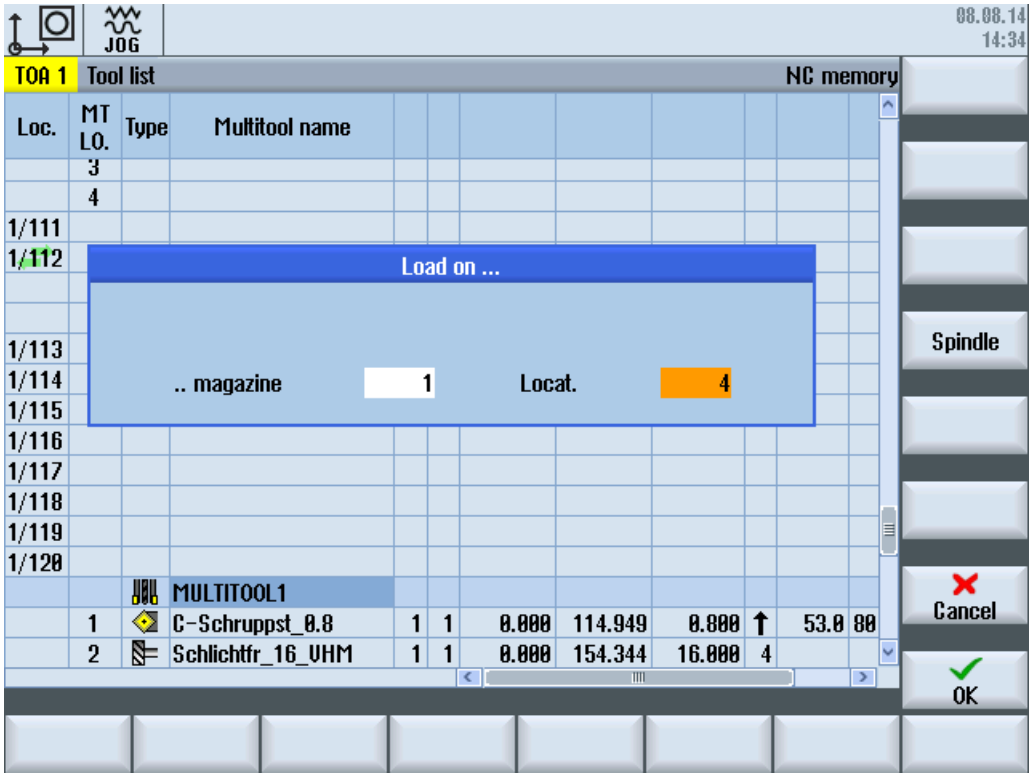



Figure 9-3 Loading dialog for a multitool

The following table shows the task status and the preparation of the acknowledgment without the position specification for the loading.

| Data block | Meaning | Value |
|---|---|---|
| DB4100.DBB0 | Task | **1**0000001 |
| DB4100.DBB1 | Task description | 00000001 |
| DB4100.DBW6 | Source magazine number | 9999 |
| DB4100.DBW8 | Source location number | 1 |
| DB4100.DBW10 | Target magazine number | 1 |
| DB4100.DBW12 | Target location number | 4 |
| DB4100.DBW56 | Multitool type | **1** |
| DB4100.DBW58 | Multitool location count | **0** |
| DB4100.DBD60 | Multitool distance | **0.0000** |
| DB4100.DBW64 | Multitool number | **0** |
| DB4100.DBW66 | Multitool location number | **0** |
| DB4100.DBW68 | Toolholder number | **0** |
| | | |
| DB4000.DBB0 | Acknowledgment | **00000001** |
| DB4000.DBW4 | Multitool location number | **0** |

In addition to the "normal" task data, the following are set:

- DB4100.DBX0.7: Data is present in the extended area. The tool is a multitool.
- DB4100.DBW56: This data item contains the type of the distance coding (= 1, = location).

All other data in the extended area is "0". Further NC information is not present. The acknowledgment is made with the total acknowledgment.

### References

A detailed description as well as an example for positioning a multitool is provided in the following literature:
SINUMERIK 840D sl Tool Management Function Manual.

## 9.7.3 Tool change for a multitool

### Tool call with T=<tool name>

Programming tool change commands remains unchanged. The programming in the NC program is called:
`T="ROUGHING_TOOL_2"`

This tool call selects an appropriate tool and loads it (turning technology).

If the tool found is located in a multitool, this does not play any role regarding activating the offset. The multitool is only involved when the data for the tool change command has been compiled. Additional data is added to the PLC command (multitool location number, multitool location distance). The PLC program must appropriately initiate the machine so that the active tool in the multitool can do the machining.

An alarm is output if the name of a multitool is programmed.

### Example

The following table shows the task status and the preparation of the acknowledgment without the position specification for the change.

| Data block | Meaning | Value |
|---|---|---|
| DB4300.DBB0 | Task | 10000001 |
| DB4300.DBB1 | Task description | 00010110 |
| DB4300.DBW6 | Source magazine number | 1 |
| DB4300.DBW8 | Source location number | 4 |
| DB4300.DBW10 | Target magazine number | 9998 |
| DB4300.DBW12 | Target location number | 1 |
| DB4300.DBW56 | Multitool type | 1 |
| DB4300.DBW58 | Multitool location count | 2 |
| DB4300.DBD60 | Multitool distance | 2.0000 |
| DB4300.DBW64 | Source multitool number | 2 |

| Data block | Meaning | Value |
|---|---|---|
| DB4300.DBW66 | Source multitool location number | **2** |
| DB4300.DBW68 | Target multitool number | **0** |
| DB4300.DBW70 | Target multitool location number | **0** |
| | | |
| DB4200.DBB0 | Acknowledgment | **00000001** |
| DB4200.DBW4 | Multitool location number | **0** |

In addition to the "normal" task data, the following are set:

| Data block | Meaning |
|---|---|
| DB4100.DBX0.7 | Data is present in the extended area. The tool is a multitool. |
| DB4100.DBW56 | Type of the distance coding (= 1, = location) |
| DB4300.DBW58 | Multitool location count |
| DB4300.DBD60 | Multitool distance, the angle for angular coding |
| DB4300.DBW64 | Multitool number of the new tool |
| DB4300.DBW66 | Multitool location number of the new tool |
| DB4300:DBW68<br>DB4300:DBW70 | Data of the old tool, if present |

The acknowledgment is made with the total acknowledgment.

## Tool call with T=<magazine location number>

For the programming of the multitool location number in the magazine, you must program the multitool location number with "MTL = p" (p = location number). The programmed value for MTL must correspond to an existing location number in the selected multitool. If not, an alarm is triggered.

The programming in the NC program is called:

```
T4 MTL=1
```

## Example

The following table shows the task status and the preparation of the acknowledgment with the position specification for the change.

| Data block | Meaning | Value |
|---|---|---|
| DB4300.DBB0 | Task | **1**0000001 |
| DB4300.DBB1 | Task description | 00010110 |
| DB4300.DBW6 | Source magazine number | 1 |
| DB4300.DBW8 | Source location number | 4 |
| DB4300.DBW10 | Target magazine number | 9998 |
| DB4300.DBW12 | Target location number | 1 |
| DB4300.DBW56 | Multitool type | **1** |
| DB4300.DBW58 | Multitool location count | **2** |
| DB4300.DBD60 | Multitool distance | **1.0000** |

| Data block | Meaning | Value |
|---|---|---|
| DB4300.DBW64 | Source multitool number | 2 |
| DB4300.DBW66 | Source multitool location number | 1 |
| DB4300.DBW68 | Target multitool number | 0 |
| DB4300.DBW70 | Target multitool location number | 0 |
| | | |
| DB4200.DBB0 | Acknowledgment | 00000001 |
| DB4200.DBW4 | Multitool location number | 0 |

In a replacement cycle for T, $C_MTL_PROG can read whether MTL was programmed. The programmed value for MTL can be read with $C_MTL .

The acknowledgment is made with the total acknowledgment.

DB4200.DBW4: For the acknowledgment, the actual position of the multitool on the NC can be notified in this data item. If the position does not match the position requested in the NC task, an alarm is triggered. If the position is not notified to the NC (DB4200.DBW4 = 0), no check of the correct multitool position is made. Value "0" is permitted only for the total acknowledgment.

# 9.8 Editor for the tool and magazine management

## 9.8.1 Overview

The tool management offers the possibility to define all real tool magazines and internal magazines with the appropriate properties from the operating software and link them channel-specific with the tool unit. You can configure the relationships between tool magazine, griper and spindle via a tree view.

### General sequence

The following steps are required to create a tool management:

- Create tool unit
- Define buffer locations
- Define loading location
- Create magazine configuration
- Link assignments

**Note**

When you create a new tool unit, a buffer location with a spindle location, a loading magazine with a loading location and a real magazine with five locations are created by default.

### Meaning of the symbols

| Symbol | Description |
|---|---|
|  | Folder for the actively used tool units |
|  | Folder with access to templates |
|  | Tool spindle / tool holder |
|  | Tool gripper 1 |
|  | Tool gripper 2 |
|  | Shows the changes in the the "Active data" folder that has not been activated yet |

### Meaning of the tree view

The following folders are available in the tree view:

- Active data: Configure and activate the magazine management.
- Examples: Copy templates into the "Active data" folder.
- Network/USB: Copy the new magazine managements into the "Active data" folder or use existing ones for other systems.

## 9.8.2 Calling the tool management

You open the tool management window as follows:

### Procedure




1. Select the "Start-up" operating area.







2. Press the "NC" and "Tool management" softkeys.

   The "Tool management" window opens.

## 9.8.3 Editing the tool management

For the tree view of the tool management, functions are available with which you can parameterize magazine data items and link them with each other.

**Note**

If you position the cursor on an element, only those functions possible for this element are active. All other functions are inactive which is indicated with grayed-out softkeys.

### Procedure




The "Tool management" window is open.

#### Adding a new tool unit

1. Position the cursor on the "Active data" folder element.




2. Press the "New element" softkey.
   A folder element with the standard designation <tool unit> is inserted and the "TO unit" input window opens.
3. You can make changes in the input window of the folder element.




4. Press the "OK" softkey to insert the folder element with the settings.

#### Inserting a new magazine

1. Position the cursor on the "Active data" folder element.



2. Navigate with the <Cursor right> and <Cursor down> keys to the folder element of the tool unit.






3. Press the "New element" softkey.
   A folder element with the standard designation <magazine> is inserted and the "Magazine" input window opens.
4. You can make changes in the input window of the folder element.




5. Press the "OK" softkey to insert the folder element with the settings.

#### Editing an element

1. Position the cursor on the "Active data" folder element.



2. Navigate with the <Cursor right> and <Cursor down> keys to the element that you want to edit.





3. Press the <Cursor right> or <INPUT> key.
   The input window for the element settings opens.







4. Enter the desired changes and press the "OK" softkey to confirm the entries.

**Note**

You define the number of buffer and magazine locations in the input window of the higher-level <Magazine> folder element with the "Locations" parameter.

### Activating Data




1. Press the "Activate data" softkey.
   All changed parameterizations are transferred to the NC and activated.
   All of the error messages are displayed again, and you have the option of specifically resolving these by modifying the data.

**Note**

Once you have made changes to the tool management, the data is marked with "*". This data is not yet in the NC and so does not have any effect.

If you want to create a tool management that is similar to an already existing element, then you can save time if you copy the existing elements and insert them at a different position. You have the option, for example of also copying inactive subtrees into the active tree.

### Copying/cutting and pasting elements

1. Position the cursor on the desired element.




2. Press the "Copy" softkey.

   - OR -




   Press the "Cut" softkey.

Paste

3. Position the cursor at the required position and press the "Paste" softkey.
   If the element exists already at the desired location, you receive further insertion options.

OK

4. Press the "OK" softkey if you wish to attach the element to the tree view as the next element.
   The element is inserted.
   **Note:**
   You have the option of changing the name of the element in the associated input window.

**Deleting elements**

Delete element

1. Place the cursor on the desired element and press the "Delete element" softkey.

OK

2. Press the "OK" softkey to confirm the deletion.
   The selected object is removed.

## 9.8.4 Parameterizing the tool management

### 9.8.4.1 Create tool unit

A tool unit is the grouping of tools and magazines to form a unit. A tool unit can be assigned one or more channels. The tools and magazines must be unique within a tool unit. For a tool unit, specify the search parameters and the channel in the "TO unit" window.

### Directory values

You are currently located under the "Active data" → <tool unit> directory in the "Commissioning" → "NC" →"Tool management" operating area.

You can set the following parameters in the "TO unit" window from the tool unit directory:

| Parameters | Description |
|---|---|
| Name | Designation that is displayed in the tree structure. |
| Channel number | The channel to which the tool unit is assigned. |

| Parameters | Description |
|---|---|
| Tool search | Search strategy:<br>• Without search strategy<br>• Active tool<br>• Shortest way |
| Empty location search | Search strategy:<br>• Without search strategy<br>• First location forwards<br>• Current location forwards<br>• Current location backwards<br>• Last location backwards<br>• Symmetric current location<br>• 1:1 exchange |

### 9.8.4.2 Define buffer locations

Buffer locations are spindles, grippers, loaders and transfer locations. All buffer locations are managed in an internal buffer magazine with the number 9998. The number of buffer locations is defined from the "Magazine" window. The buffer location type is defined from the "Location" window.

#### Directory values

You are currently located under the "Active data" → <tool unit> → <buffer> directory in the "Commissioning" → "NC" →"Tool management" operating area.

You can define the following parameters in the "Magazine" window from the buffer directory:

| Parameters | Description |
|---|---|
| Name | Designation of the buffer magazine<br>The designation is displayed in the tree structure. |
| Magazine number | 9998 (cannot be changed) |
| Locations | Number of the buffer locations |

#### Element values

You can change the buffer location type with the parameters in the "Location" window:

| Parameters | Description |
|---|---|
| (Selection menu) | Select a buffer type<br>• Tool spindle / tool holder<br>• Loader<br>• Gripper<br>• Transfer location |
| Index | The index counts the locations of a type. |

### 9.8.4.3 Define loading location

Loading location are loading points or loading stations. All loading locations are stored in an internal loading magazine with the number 9999. The number of loading locations is defined from the "Magazine" window. The loading location type is defined from the "Location" window.

#### Directory values

You are currently located under the "Active data" → <tool unit> → <loading magazine> directory in the "Commissioning" → "NC" →"Tool management" operating area.

You can define the following parameters in the "Magazine" window from the loading magazine directory:

| Parameters | Description |
|---|---|
| Name | Designation of the loading magazine<br>The designation is displayed in the tree structure. |
| Magazine number | 9999 (cannot be changed) |
| Locations | Number of loading locations |

#### Element values

You can change the loading location type with the parameters in the "Location" window:

| Parameters | Description |
|---|---|
| (Selection menu) | Select a loading location type:<br>• Load location<br>Load locations are areas on the machine at which the direct loading in the magazine is possible. The tool can be inserted directly into the magazine at this location. The magazine location to be loaded is moved to the loading point. For example, chain magazines have loading points.<br>• Loading station<br>A loading station is a location outside the magazine onto which the tool to be loaded is placed. The tool is then transported from that location to the magazine via a transport mechanism. Loading stations are generally used for box or chain magazines. |
| Index | The index counts the locations of a type. |

**Note**

Loading point 1 is always set automatically as loading/unloading point for spindle 1 (manual loading point).

### 9.8.4.4 Create magazine configuration

You can define several magazines in a tool unit. You define the magazine configuration via the "Magazine" window. The magazine location type is defined from the "Location" window.

## Directory values

You are currently located under the "Active data" → <tool unit> → <magazine> directory in the "Commissioning" → "NC" →"Tool management" operating area.

You can define the following parameters in the "Magazine" window from the magazine directory:

| Parameters | Description |
|---|---|
| Magazine type | Selection menu:<br>• Chain magazine<br>• Box magazine<br>• Revolver |
| Name | Designation of the magazine.<br>The designation is displayed in the tree structure. |
| Magazine number | Specifies the number of the magazine |
| Locations | Number of magazine locations |
| Tool search | Search strategy:<br>• Without search strategy<br>• Take from the TOA<br>• Active tool<br>• Shortest way |
| Empty location search | Search strategy:<br>• Without search strategy<br>• Take from the TOA<br>• First location forwards<br>• Current location forwards<br>• Current location backwards<br>• Last location backwards<br>• Symmetric current location<br>• 1:1 exchange |
| Location coding | • Variable<br>• Fixed-location coded |
| Location type | Identification for the magazine location type<br>(Changes the value globally for all locations in the magazine) |
| Consider adjacent location | • Yes<br>• No<br>(Changes the value globally for all locations in the magazine) |

**Note**

If one of the Location type or Consider adjacent location parameters is changed individually and the value is not identical for all locations, the "Variable" definition appears in the display.

### Element values

You can change the magazine location type with the parameters in the "Location" window:

| Parameters | Description |
|---|---|
| Location type | Identification for the magazine location type |
| Consider adjacent location | • Yes<br>• No |

## 9.8.4.5 Link assignments

You can establish the mechanical relationship between the individual components of the tool unit in the "Assignment" window.

### Element values

You are currently located under the "Active data" → <tool unit> directory in the "Commissioning" → "NC" →"Tool management" operating area.

You can link the following parameters with each other from the selection list via the Assignments element:

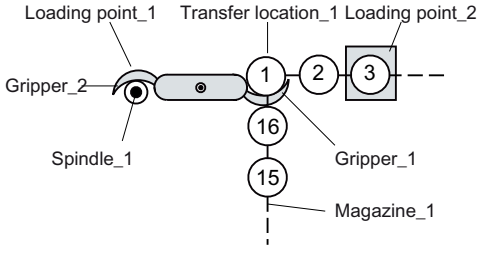
- Spindle/gripper/loader for magazine
- Gripper/loader for spindle
- Loading point for magazine

**Note**

The separation as number of locations from the magazine zero point of the affected magazine to the internal magazine location must be defined for each magazine assignment.

### Example

If, for example, tools on a spindle must be replaced from a magazine, the following assignment can result:



| Parameters | Assignment |
|---|---|
| Spindle/gripper/loader for magazine | • Tool spindle location 1 for magazine (separation 0)<br>• Gripper location 2 for magazine (separation 0)<br>• Gripper location 3 for magazine (separation 0)<br>• Transfer location 4 for magazine (separation 0) |
| Gripper/loader for spindle | • Tool spindle location 1 for gripper 2<br>• Tool spindle location 1 for gripper 3 |
| Loading point for magazine | • Loading point location 1 for magazine (separation 0)<br>• Loading point location 2 for magazine (separation 3) |

# Service and diagnostics 10

## 10.1 NC/PLC variables

### 10.1.1 Displaying and editing PLC and NC variables

The "NC/PLC Variables" window allows NC system variables and PLC variables to be monitored and changed.

You receive the following list in which you can enter the desired NC/PLC variables in order to display the actual values.

- Variable
  Address for NC/PLC variable.
  Incorrect variables have a red background and are displayed with a # character in the value column.
- Comment
  Any comment on the variable.
  The columns can be displayed and hidden.
- Format
  Specify the format in which the variable is to be displayed.
  The format can be specified (e.g. floating point).
- Value
  Displays the actual value of the NC/PLC variables.

| PLC variables | |
|---|---|
| Inputs | Input bit (Ex), input byte (EBx), input word (EWx), input double word (EDx) |
| Outputs | Output bit (Ax), output byte (ABx), output word (AWx), output double word (ADx) |
| Bit memory | Memory bit (Mx), memory byte (MBx), memory word (MWx), memory double word (MDx) |
| Times | Time (Tx) |
| Counters | Counter (Cx) |
| Data | Data block (DBx): Data bit (DBXx), data byte (DBBx), data word (DBWx), data double word (DBDx) |

| Formats | |
|---|---|
| B | Binary |
| H | Hexadecimal |
| D | Decimal without sign |
| +/-D | Decimal with sign |

| Formats | |
|---|---|
| F | Floating point (for double words) |
| A | ASCII character |

## Notation for variables

- PLC variables
  EB2
  A1.2
  DB2.DBW2
- NC variables
  - NC system variables - notation
    $AA_IM[1]
  - User variables/GUDs - notation
    GUD/MyVariable[1,3]
  - OPI - notation
    /CHANNEL/PARAMETER/R[u1,2]

**Note**

**NC system variables and PLC variables**

- System variables can be dependent on the channel. When the channel is switched over, the values from the corresponding channel are displayed.
- For user variables (GUDs) it is not necessary to make a specification according to global or channel-specific GUDs. The indices of GUD arrays are, just like NC variables in the system variable syntax, 0-based; this means that the first element starts with the index 0.
- Using the tooltip, for NC system variables, you can display the OPI notation (with the exception of GUDs).

### Changing PLC variables

Changes can only be made to the PLC variables with the appropriate password.

**⚠ WARNING**

**Incorrect parameterization**

Changes in the states of NC/PLC variables have a major impact on the machine. Incorrect configuration of the parameters can endanger human life and cause damage to the machine.

## Changing and deleting values




1. Select the "Diagnostics" operating area.




2. Press the "NC/PLC variables" softkey.

   The "NC/PLC Variables" window opens.

3. Position the cursor in the "Variable" column and enter the required variable.




4. Press the <INPUT> key.

   The operand is displayed with the value.




5. Press the "Details" softkey.

   The "NC/PLC Variables: Details" window opens. The information for "Variable", "Comment" and "Value" is displayed in full length.




6. Position the cursor in the "Format" field and choose the required format with <SELECT>.




7. Press the "Display comments" softkey.

   The "Comments" column is displayed. You have the option of creating comments or editing existing comments.




   Press the "Display comments" softkey once again to hide the column.




8. Press the "Change" softkey if you would like to edit the value.

   The "Value" column can be edited.




9. Press the "Insert variable" softkey if you wish to select a variable from a list of all existing variables and insert this.

   The "Select Variable" window opens.




10. Press the "Filter/search" softkey to restrict the display of variables (e.g. to mode group variables) using the "Filter" selection box and/or select the desired variable using the "Search" input box.




   Press the "Delete all" softkey if you would like to delete the entries for the operands.




11. Press the "OK" softkey to confirm the changes or the deletion.

    - OR -




    Press the "Cancel" softkey to cancel the changes.

**Note**

**"Filter/Search" when inserting variables**

The start value for "Filter/Search" of variables differs.

For example, to insert the variable $R[0], set "Filter/Search":

- The start value is 0, if you filter according to "System variables".
- The start value is 1, if you filter according to "All (no filter)". In this case, all signals are displayed and shown in the OPI notation.

### Changing operands

Depending on the type of operand, you can increment or decrement the address by 1 place at a time using the "Operand +" and "Operand -" softkeys.

**Note**

**Axis names as index**

For axis names, the softkeys "Operand +" and "Operand -" do not act as index, e.g. for $AA_IM[X1].




**Examples**

DB97.DBX2.5
Result: DB97.DBX2.6
$AA_IM[1]
Result: $AA_IM[2]




MB201
Result: MB200
/Channel/Parameter/R[u1,3]
Result: /Channel/Parameter/R[u1,2]

## 10.1.2 Saving and loading screen forms

You have the option of saving the configurations of the variables made in the "NC/PLC variables" window in a screen form that you reload again when required.

**Editing screen forms**

If you change a screen form that has been loaded, then this is marked using with * after the screen form name.

The name of a screen form is kept in the display after switching-off.

### Procedure

1. You have entered values for the desired variables in the "NC/PLC variables" window.



2. Press the ">>" softkey.




3. Press the "Save screen" softkey.

   The "Save screen: Select archiving" window opens.




4. Position the cursor on the template folder for variable screen forms in which your actual screen form should be saved and press the "OK" softkey.

   The "Save screen: Name" window opens.




5. Enter the name for the file and press the "OK" softkey.

   A message in the status line informs you that the screen form was saved in the specified folder.

   If a file with the same name already exists, they you will receive a prompt.




6. Press the "Load screen" softkey.

   The "Load screen" window opens and displays the sample folder for the variable screen forms.

7. Select the desired file and press the "OK" softkey.

   You return to the variable view. The list of all of the predefined NC and PLC variables is displayed.

## 10.2 Displaying the Service overview

The operating state of machine axes and drives is displayed in the "Service Overview" window. You can configure the display via the "Extended selection >" softkey or directly via the drop-down list according to the following criteria:

- All NC axes and drives (= default)
- NC axes
- Drives without NC axis assignment

### Enable displays

- On/Off1
- Off2
- Off3
- From the drive: Operation enabled
- From the infeed: Enable operation
- Enable pulses
- Speed controller enable NC

- Enable pulses
- Drive ready
- Heat sink temperature
- Power section in $i^2t$ limiting
- Motor temperature
- **Measuring system 1 active / measuring system 2 active**
  The states for measuring system 1/2 have the following meaning:

| Symbol | Meaning |
|---|---|
|  | The position measuring system is active. |
|  | The position measuring system is parked.<br>- OR - -<br>The position measuring system is not configured. |
|  | The position measuring system is passive. |

## Procedure




1. Select the "Diagnostics" operating area.



2. Press the menu forward key.
   New horizontal softkeys are displayed.




3. Press the "Axis diagnostics" softkey.
   The "Service Overview" window opens.
4. Select an overview in the combo box "Select" combo box.
   - OR -




   Press the "Go to selection" softkey to search for an overview.

## 10.2.1 Selecting axes and drives

To display certain enable signals and statuses of machine axes, you can arrange a selection of all the axes and drives that are available in any sequence you wish.

**Note**

The following selection configurations have been previously defined, and cannot be changed or deleted:

- All NC axes and drives
- NC axes
- Drives without NC axis assignment

### Selection overview

The following information is provided in the selection overview.

| Column | Description |
|---|---|
| Bus No. | Bus number for drives without NC assignment.<br>No entry for NC axes. |
| Slave address | Slave address for drives without NC assignment.<br>No entry for NC axes. |
| Device No. | Device number for drives without NC assignment.<br>No entry for NC axes. |
| Drive object No. | Drive object number for drives without NC assignment.<br>No entry for NC axes. |
| Name/identifier | For NC axes contains the standard machine axis identifier and the user-defined machine axis names.<br>For drives without NC assignment, contains the drive object names. |
| Assignment | NC is displayed for an internal object assignment.<br>PLC is displayed for an external object assignment. |

**Note**

The selection overview differs in the information provided regarding the slave address and device number between PROFIBUS and PROFINET configuration.

### Procedure




1. Select the "Diagnostics" operating area.



2. Press the menu forward key.
   New horizontal softkeys are displayed.




3. Press the "Axis diagn." softkey.
   The "Service Overview" window opens.




4. Press the "Extended selection" softkey.
   The window "Extended selection: Axes and drives" opens.




5. Press the "New selection" softkey.
   The "New selection" dialog is opened.
6. Enter a display and a file name in the entry fields.




7. Confirm the entry with the "OK" softkey.
   The "Edit selection" window is opened for the newly created selection.
8. Set checkmarks in the check boxes for the axes and drives that you wish to display.
   - OR -
   Press one of the following softkeys to obtain a certain selection:




All axes/drives are selected.




No axis/drive is selected.




Only internal drives are selected.




Only external drives are selected.




Those axes are selected that are assigned to a real drive.




All axes are selected, which are defined in a minimum of one channel.




9. Press the "OK" softkey to confirm your selection.




10. Press the "Delete selection" softkey to delete the actual selection.




11. Press the "Edit selection" softkey to edit the actual selection.
    The "Edit selection" window opens.




12. If you click the "Back" softkey, you will return to the "Service Overview" main screen.

## 10.2.2 Axis diagnostics

### Application

The following information is displayed in the "Service Overview" window:

- Check of the setpoint branch (e.g. programmed position setpoint, speed setpoint, spindle speed setpoint)
- Check of the actual value branch (e.g. actual position value, measuring system 1 or 2, actual speed value)
- Optimization of the position control loop of the axis (e.g. following error, control deviation, servo gain factor)
- Check of the entire control loop of the axis (e.g. through position setpoint/actual-value comparison and speed setpoint/actual-value comparison)
- Check of hardware errors (e.g. by checking the encoder: If the axis is moved mechanically, the actual position value must change)
- Setting and check of the axis monitoring functions.

### Procedure




1. Select the "Diagnostics" operating area.



2. Press the menu forward key.
   New horizontal softkeys are displayed.




3. Press the "Axis diagnostics" softkey.
   The "Service Overview" window opens.




4. Press the "Service axis" softkey.
   The "Service Axis/Spindle" window opens.







5. Press the "Axis +" or "Axis -" softkey to scroll the axes forward or backward.

- OR -




Press the "Axis selection" softkey.
The "Axis Direct Selection" window opens.
Select the required axis directly from those available in the drop-down list box.




6. Confirm the selection with "OK".
   The values of the axis are displayed.

## Display data

| Display data | Meaning |
|---|---|
| Following error | The difference between the position setpoint and the actual position value of the active measuring system 1 or 2.<br>Unit: mm, inch or degrees |
| System deviation | The difference between the position setpoint at the position controller input and the actual position value of the active measuring system 1 or 2.<br>Unit: mm, inch or degrees |
| Contour deviation (axial) | The actual contour deviation is displayed with this value (variations of the following error caused by equalization operations on the speed controller due to load changes).<br>The contour deviation results from the difference between an actual position pre-calculated from the position setpoint and the actual position value of active measuring system 1 or 2.<br>Unit: mm, inch or degrees |
| Servo gain factor (calculated) | The servo gain factor in the display is calculated by the NC according to the following equation:<br>$K_v$ - factor = $\frac{\text{Speed setpoint}}{\text{Following error}}$ ;<br>Unit (for default setting): $\frac{[\text{m/min}]}{[\text{mm}]}$ ;<br>Velocity setpoint = setpoint currently being output to the axis/spindle<br>**References:**<br>Basic Functions Function Manual; Velocities, Setpoint / Actual Value Systems, Closed-Loop Control (G2) |
| Active measuring system | **0:** No measuring system active.<br>**1:** Measuring system 1 active.<br>**2:** Measuring system 2 active. |
| State of measuring system 1<br>State of measuring system 2 | The state of measuring system 1 or 2 is output here:<br>Active - Parked - Passive |
| Actual position value measuring system 1<br>Actual position value measuring system 2 | The actual position of the axis measured via measuring system 1 or 2.<br>The position is displayed in the machine coordinate system (no work offsets or tool offsets taken into account).<br>Unit: mm, inch or degrees |
| Position setpoint | Specified position transferred from the interpolator to the position control<br>Unit: mm, inch or degrees |
| Absolute compensation value measuring system 1<br>Absolute compensation value measuring system 2 | Display of the absolute compensation value for measuring system 1 or 2.<br>The compensation value consists of the sum of backlash and leadscrew error compensation for the actual axis position.<br>Unit: mm, inch or degrees |

| Display data | Meaning |
|---|---|
| Compensation, sag + temperature | Display of the compensation value calculated for the current axis position based on the total of the sag and temperature compensations.<br>Unit: mm, inch or degrees |
| Actual speed value, active encoder | The pulses supplied by the encoder are evaluated by the NC and displayed.<br>Unit: %<br>100% means maximum speed. |
| Speed setpoint, drive | Speed setpoint transferred to the drive (= speed setpoint from position controller and feedforward control).<br>Unit: %<br>100% means maximum speed setpoint. |
| Programmed spindle speed setpoint | Speed setpoint programmed by the user.<br>Unit: rpm<br>For example: Input: S1000; display: 1000 rpm<br>Display applies to spindles only. |
| Spindle speed setpoint current | Current active speed setpoint with correct sign, including calculated compensation value and any active speed limitation (specified by setting or machine data)<br>Unit: rpm<br>Display applies to spindles only. |
| Position offset to the leading axis / spindle actual value | The currently valid position offset value is displayed here (relative to the actual value) if a position offset (angular offset between the following and leading spindle) has been programmed within the synchronous spindle functionality.<br>Unit: mm, inches, degrees<br>**References:**<br>Function Manual, Extended Functions; Synchronous Spindle (S3) |
| Position offset to the leading axis / spindle setpoint | The currently valid position offset value is displayed here (relative to the setpoint) if a position offset (angular offset between the following and leading spindle) has been programmed within the synchronous spindle functionality.<br>Unit: mm, inches, degrees<br>**References:**<br>Function Manual, Extended Functions; Synchronous Spindle (S3) |
| Override | The effective correction factor of the feed or spindle correction switch is displayed.<br>Unit: % |
| Current gear stage | Display of the current actual gear stage<br>With axes, this is only displayed if a spindle is assigned to the axis. The display corresponds to the NC/PLC interface signal:<br>DB31, ... DBX16.0-2 (actual gear stage)<br>**References**:<br>Function Manual, Basic Functions; Spindles (S1) |

| Display data | Meaning |
|---|---|
| Parameter set (axis) | Displays which of the six parameter sets of the position controller is active.<br>**References**:<br>Basic Functions Function Manual; Velocities, Setpoint / Actual Value Systems, Closed-Loop Control (G2) |
| Controller mode | Display of the current controller state:<br>• Position control<br>• Speed control<br>• Hold<br>• Parking<br>• Follow-up<br>• Braking<br>**References:**<br>Function Manual, Basic Functions; Various NC/PLC interface signals and functions (A2) |
| Feedforward control mode | Indicates whether and if so, which mode of the dynamic feedforward control for the axis is active:<br>• Inactive<br>• Velocity<br>The velocity-dependent speed feedforward control is active.<br>• Torque<br>The acceleration-dependent torque feedforward control is active (in combination with the speed feedforward control)<br>**References:**<br>Extended Functions Function Manual; Compensations (K3) |
| "Referenced" state | Status display for reference point approach (axis):<br>• Active MS need not be referenced<br>• Active MS referenced<br>• Active MS must be referenced<br>The display depends on the settings in the machine data:<br>• MD34110 $MA_REFP_CYCLE_NR<br>• MD20700 $MC_REFP_NC_START_LOCK<br>The display corresponds to the NC/PLC interface signal:<br>DB31, ... DBX60.4 and 60.5 (referenced/synchronized 1 or 2)<br>**References:**<br>Function Manual Basic Functions; Reference Point Travel (R1) |

| Display data | Meaning |
|---|---|
| QEC state | Indicates whether and if so, which method of the quadrant error compensation (QEC) is active for the axis:<br>• Inactive<br>• Neural QEC learning active<br>• Conventional QEC active<br>• Conventional QEC w. adapt. comp. value act.<br>(conventional QEC with adaptation of the compensation value active)<br>• Neural QEC active<br>• Neural QEC w. ad. meas. time val. act.<br>(neural QEC active with adaptation of the measuring time active)<br>• Neural QEC w. ad. dec. time comp. val. act.<br>• (neural QEC with adaptation of decay time of compensation value active)<br>• Neural QEC w. ad. meas. time + dec. time comp. val. act.<br>(neural QEC active with adaptation of measuring time and decay time of compensation value active)<br>**References:**<br>Extended Functions Function Manual; Compensations (K3) |
| "Travel to fixed stop" state | Indicates whether or not the axis has fulfilled the conditions for "Fixed stop reached" when the "Travel to fixed stop" function is active (IS DB31, ... DBX62.5):<br>• Normal control<br>("Travel to fixed stop" function not activated)<br>• Fixed stop reached<br>• Failed<br>**References:**<br>Function Manual, Basic Functions; Travel to Fixed Stop (F1) |
| Torque limitation value | Indicates the value programmed via FXST[x] or SD43510 $SA_FIXED_STOP_TORQUE or the value defined via MD37010 $MA_FIXED_STOP_TORQUE_DEF for the clamping torque for "Travel to fixed stop".<br>Unit: % of maximum torque<br>**References:**<br>Function Manual, Basic Functions; Travel to Fixed Stop (F1) |

## 10.2.3 Service drive

### Display drive and motor data

The "Service Drive" window displays important information about the status of the motors and drive modules, such as motor temperature and DC-link voltage.

## Procedure




1. Select the "Diagnostics" operating area.



2. Press the menu forward key.
   A switch is made to a new softkey bar.




3. Press the "Axis diagnostics" softkey.
   The "Service Overview" window opens.




4. Press the "Service drive" softkey.
   The "Service Drive" window opens.







5. Press the "Drive +" or "Drive -" softkey to scroll forward or backward.

   - OR -




   Press the "Drive selection" softkey.
   The "Drive Direct Selection:" window opens.
   Select the desired drive directly from the selection list.




6. Confirm the selection with "OK."
   The drive data is displayed.

## Overview

The individual status displays, warnings, messages, etc. that are displayed in the "Service Drive" window are explained in the following.

## PLC pulse enable

The display, whether the pulse enable from the PLC is available for the drive corresponds to the interface signal:
DB380x, ... DBX4001.7 "Pulse enable".

| Status | Meaning | Display |
|---|---|---|
| 1 | Pulse enable for this drive is activated by the PLC. | Yes |
| 0 | The pulses for the drive module are disabled from the PLC. | No |

### Speed controller enable NC

The display, whether the speed controller enable from the NC is available for the drive corresponds to the interface signal:
DB390x ... DBX1.6 "Speed controller active".

| Status | Meaning | Display |
|---|---|---|
| 1 | Speed controller enable from the NC present. | Yes |
| 0 | Speed controller enable from the NC not present. | No |

### Rampup function generator quick stop

The status display for the ramp-function generator quick stop corresponds to the interface signal:
DB3900 ... DBX4000.1 "Ramp-function generator disable active".

| Status | Meaning | Display |
|---|---|---|
| 1 | Ramp-up function generator quick stop is active. The drive is stopped without a ramp function with speed setpoint = 0 and without pulse suppression. | Yes |
| 0 | Ramp-up function generator quick stop is not active for the drive. | No |

### Pulses enabled

The message whether the pulses have been enabled for the drive corresponds to the interface signal:
DB390x, ... DBX4001.7 "Pulses enabled".

| Status | Meaning | Display |
|---|---|---|
| 1 | The drive module pulses are enabled. The axis/spindle can now be traversed. | Yes |
| 0 | The drive module pulses are suppressed. The axis/spindle can therefore not be traversed. | No |

### Drive ready

The display of the current status of the selected drive corresponds to the interface signal:
DB390x, ... DBX4001.5 "Drive ready".

| Status | Meaning | Display |
|---|---|---|
| 1 | The drive is ready. | Yes |
| 0 | The drive is not ready. | No |

### Ramp-up phase

The display of the current ramp-up phase of the selected drive corresponds to the drive parameter:
r0002 "Drive operating display".

### Number of faulty signs-of-life

Display of communications errors detected in hardware between NC and drive.

**Note**

If the display shows a value other than "0", please contact your Siemens regional office!

### SC1 drive alarm message

Displays (yes/no) whether messages of status class 1 are active. Status class 1 messages are alarms with the following properties:

- They lead to internal responses (e.g. regenerative braking, immediate pulse suppression)
- They are modal.

This is a group message. For details on the actually pending drive alarms, please refer to the "Drive System Diagnostics" window, see Service drive (Page 313).

### DC-link voltage, smoothed

The display of the smoothed actual value of the DC-link voltage of the selected drive corresponds to the drive parameter:
r0026 "DC-link voltage smoothed".

Unit: Volts

### Speed setpoint, smoothed

The display of the smoothed speed setpoint - measured according to the setpoint limitation for the P component of the speed controller - corresponds to the drive parameter:
r1438 "Speed controller speed setpoint".

Unit: rpm

### Actual speed value

The display of the smoothed actual value of the motor speed corresponds to the drive parameter:
r0021 "Actual speed value smoothed".

Unit: rpm

### Actual current value, smoothed

The display of the smoothed actual current value corresponds to the drive parameter:
r0078[1] "Actual current value, torque-generating".

Unit: A

### Motor temperature

The display of the current temperature in the motor corresponds to the drive parameter:
r0035 "Motor temperature"
Unit: °C

### Integrator disabling

The display as to whether the integrator of the speed controller is active, corresponds to the interface signal:
DB390x, ... DBX4001.6 "n-controller integrator disabled".

| Status | Meaning | Display |
|---|---|---|
| 1 | 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. | Yes |
| 0 | The integrator of the speed controller is enabled. The speed controller functions as a PI controller. | No |

### Parking axis

Displays (yes/no) whether it is a parking axis/spindle.

**Note**

With parking axes/spindles, all encoder-specific monitoring and evaluation functions are switched off. This allows the encoder to be withdrawn without initiating an alarm.

### Specified drive data set

The display (standard: DDS0) of which of the eight drive parameter sets is to be activated by the PLC corresponds to the interface signal:
DB380x, ... DBX4001.0 to 4001.2 "Drive parameter set selection A, B, C".

### Actual drive data set

The display (standard: DDS0) of which of the eight drive parameter sets is currently active corresponds to the interface signal:
DB390x, ... DBX4001.0 to 4001.2 "Active drive parameter set A, B, C".

### Specified motor data set

The display (MDS0...3) of which motor data set is to be activated by the PLC corresponds to the interface signal:
DB380x, ... DBX4001.3 to 4001.4 "Motor selection A, B".

The following assignment applies:

| Motor data set | Coding | |
|---|---|---|
| MDS0 | 0 | 0 |
| MDS1 | 0 | 1 |
| MDS2 | 1 | 0 |
| MDS3 | 1 | 1 |

## Actual motor data set

The display (MDS0...3) of which motor data set is currently active corresponds to the interface signal:
DB390x, ... DBX4001.3 to 4001.4 "Active motor A, B".

## Operating mode

The display of the control type of a drive corresponds to the drive parameter:
p1300[0…n] "Open-loop/closed-loop control mode".

Depending on "n", the following "values" are displayed:

| n | Display |
|---|---|
| 20 | Speed control (without encoder) |
| 21 | Speed control (with encoder) |
| 23 | Torque control (with encoder) |

## Position actual value measuring system 1/2

The actual position of the axis as measured via measuring system 1/2. The position is displayed in the machine coordinate system (no zero offsets or tool offsets included).

Unit: mm, inches or degrees

## Heat sink temperature

The display of whether the heat sink temperature is OK corresponds to the interface signal:
DB3390x, ... DBX4002.1 "Heat sink temperature prewarning".

| Status | Meaning | Display |
|---|---|---|
| 1 | The drive signals a "Heat sink temperature prewarning" to the PLC. | Overtemper-ature |
| 0 | The drive module heat sink temperature pre-warning has not responded. | OK |

## Motor temperature

The display of whether the motor temperature is OK corresponds to the interface signal:
DB390x, ... DBX4002.0 "Motor temperature prewarning".

| Status | Meaning | Display |
|---|---|---|
| 1 | The motor temperature has exceeded the warning threshold configured in the drive. | Overtemperature |
| 0 | The motor temperature is below the warning threshold. | OK |

## Ramp-up function completed

The status display of the drive as to whether the ramp-up has been completed corresponds to the interface signal:
DB390x, ... DBX4002.2 "Ramp-up completed".

| Status | Meaning | Display |
|---|---|---|
| 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. | Yes |
| 0 | The ramp-up procedure is still active after the speed setpoint has been changed. | No |

## Torque lower than threshold setting

The status display of the drive as to whether the threshold torque has been undershot corresponds to the interface signal:
DB390x, ... DBX4002.3 "$|M_d| < M_{dx}$".

| Status | Meaning | Display |
|---|---|---|
| 1 | The current torque utilization lies below the torque utilization threshold set in p2194. The drive signals the PLC that the torque setpoint $|M_d|$ does not exceed the threshold torque $M_{dx}$. | Yes |
| 0 | The torque setpoint $|M_d|$ is larger than the threshold torque $M_{dx}$. This signal can be used to determine whether the motor is overloaded. | No |

## Speed lower than minimum setting

The status display of the drive as to whether the minimum speed has been undershot corresponds to the interface signal:
DB390x, ... DBX4002.4 "$|n_{act}| < n_{min}$".

| Status | Meaning | Display |
|---|---|---|
| 1 | The actual speed value $|n_{act}|$ is less than the threshold minimum speed $n_{min}$ set in p2161. | Yes |
| 0 | The actual speed value is greater than the threshold minimum speed. | No |

### Speed lower than threshold setting

The status display of the drive as to whether the threshold speed has been undershot corresponds to the interface signal:
DB390x, ... DBX4002.5 "$|n_{act}| < n_x$".

| Status | Meaning | Display |
|---|---|---|
| 1 | The actual speed value $|n_{act}|$ is less than the threshold speed $n_x$ set in p2155. | Yes |
| 0 | The actual speed value is greater than the set threshold speed. | No |

### Actual speed = set speed

The status display of the drive as to whether the actual speed value matches the speed setpoint corresponds to the interface signal:
DB390x, ... DBX4002.6 "$|n_{act}| < n_{set}$".

| Status | Meaning | Display |
|---|---|---|
| 1 | The speed deviation between the setpoint and actual value is within the tolerance specified in p2163. | Yes |
| 0 | The speed deviation between setpoint and actual value is outside the specified tolerance. | No |

### Diagnostics for alarms

This information is also provided as a diagnostic tool for diagnosing the causes of alarms such as:

- Drive fault:
  ⇒ **SC1 drive alarm message** is set.
  ⇒ Check **specified drive data set**, **specified motor data set**, **DC-link voltage**.
- Alarm 25040 "Standstill monitoring",
  Alarm 25050 "Contour monitoring",
  Alarm 25060 "Speed setpoint limitation",
  Alarm 25080 "Positioning monitoring"
  ⇒ the drive enable may be missing (**PLC pulse enable** not present); this results in the display **Pulses enabled** = No.
- **Motor temperature** = exceeded
  ⇒ check the current motor temperature.

Information about alarms and their rectification is provided in the online help.

### References

Further information about the interface signals is contained in:
NC variables and interface signals list manual

Further information about the drive parameters is contained in:
System variables list manual

Further information about the drive and motor data sets is contained in:
Base functions function manual, Chapter "A2: Various NC/PLC interface signals and functions" > "Switchover motor/drive data sets" > "Overview of the interfaces"

# 10.3 PROFINET diagnostics

## 10.3.1 Displaying the PROFINET configuration

Display the PROFINET status for diagnostic purposes during the configuration or when errors occur. This diagnostics window is only intended for information purposes. You cannot make changes there.

**PROFINET connections**

- PN1
- PN2

The PPU variant PPU 24x.3 BASIC only has one PROFINET connection (PN).

**Note**

**PROFIBUS DP connection**

The SINUMERIK 828D only has one integrated PROFIBUS DP. The NC communicates with the drives via this integrated PROFIBUS DP. It cannot be accessed from outside on the PPU and is therefore not described further here.

| Display | Meaning/information |
|---|---|
| **Status** | |
| Bus status | POWER_ON (0): Status after the control is switched on.<br>OFFLINE (1): A basic initialization has been performed.<br>STOP (2): Start in accordance with the hardware configuration (SDB).<br>CLEAR (3): The PROFINET devices have been parameterized and configured in accordance with the hardware configuration (SDB) and taken into the cyclic data exchange with zero output data.<br>OPERATE (4): The cyclic data exchange with the PROFINET devices is running.<br>ERROR (9): A fatal error had been detected (e.g. invalid or faulty SDB).<br>(<n>) Corresponds to the OPI variable value for the bus state. |
| **Bus configuration** | |
| S7 subnet ID | S7 subnet ID of the PROFINET subnet |
| Baud rate | Transmission rate in Mbd |
| Cycle time | Configured bus cycle time in ms; defines the position control cycle clock at the same time. |
| Sync. Component (TDX) | Configured interval in ms for the cyclic data exchange within a PROFINET cycle. |

| Display | Meaning/information |
|---|---|
| **PROFIBUS diagnostics/devices** | |
| Device name | Name of the PROFINET device. |
| Device no. | Number of the PROFINET device. |
| Assignment | Information as to whether the PROFINET device is assigned to the NC or to the PLC.<br>NC: e.g. one or more drives controlled by the NC.<br>PLC: e.g. I/Os or an axis controlled by the PLC, NC/PLC. |
| Communication state | Information as to whether the PROFINET device is identified on the bus via the communication state.<br>Green: The PROFINET device has been detected on the PROFINET and the data exchange is working error-free with the assigned component (NC and/or PLC).<br>Red: Incorrect response / no communication |
| Synchr. with NC | Information as to whether the PROFINET device is running synchronously to NC on the bus.<br>Green: The PROFINET is running synchronously to the NC on the PROFINET, i.e. there is an isochronous data exchange.<br>Red: Incorrect response / no communication<br>Gray: The PROFINET device is not assigned to the NC, but rather to the PLC. |
| Number of subslots | Number of configured subslots within the PROFINET device. |

## References

For further information about the network interface for PROFINET, please refer to:

- SINUMERIK 828D PPU Manual
- SINUMERIK 828D Commissioning CNC Commissioning Manual

## Procedure




1. Select the "Diagnostics" operating area.



2. Press the menu forward key and the "Bus TCP/IP" softkey.
   The "PROFINET/PROFIBUS Diagnostics" window opens.







3. Press the "Bus -" or "Bus +" softkey to select the desired bus.

...



- OR -




4. Press the "Select bus" softkey and select the desired bus in the displayed list.




5. Press the "OK" softkey.

## 10.3.2 Displaying details for PROFINET devices

Further information about the subslots of a selected PROFINET device is displayed here.

**Note**

**PROFIBUS DP connection**

The SINUMERIK 828D only has one integrated PROFIBUS DP. The NC communicates with the drives via this integrated PROFIBUS DP. It cannot be accessed from outside on the PPU and is therefore not described further here.

| Display | Meaning/information |
|---|---|
| **Device** | |
| Name | Name of the PROFINET device. |
| Number | Number of the PROFINET device. |
| IP address | IP address of the PROFINET device. |
| Assignment | Information as to whether the PROFINET device is assigned to the NC or to the PLC. |
| Communication state | Current communication state of the PROFINET connection to the device.<br>Green: The communication is active.<br>Red: The communication is not active. |
| **Slots** | |
| No. | Number of the slot. |
| Subslot no. | Number of the subslot within the device. |
| I/O addr. | I/O address in the I/O address space of the PLC assigned to this subslot.<br>For NC axes, the setpoint and the actual value must always be configured on the same I/O address. |
| Type | Specification, whether the subslot is an input, output or diagnostics slot.<br>If the subslot is assigned to an NC axis, then the output is always denoted as setpoint and the input always as actual value. |
| Length (bytes) | Length of the I/O area reserved for the subslot in the STEP7 I/O address space. |
| Communication state | Actual communication state of the subslot<br>If a value is not available for the NC axes, then the display remains hidden.<br>Green: The communication is active.<br>Red: The communication is not active.<br>Gray: No NC axis. |

| Display | Meaning/information |
|---|---|
| Machine axis | Display of the name defined in the machine data for this slot<br>If the slot is not assigned to any NC axis, then the display remains hidden. |
| Telegr. type (NC) | Displays the assigned telegram type<br>If a telegram type is not assigned in the NC machine data, then the display remains hidden. |

### Procedure




1. Select the "Diagnostics" operating area.



2. Press the menu forward key and the "Bus TCP/IP" softkey.
   The "PROFINET/PROFIBUS Diagnostics" window opens.







3. Press the "Bus -" or "Bus +" softkey to select the desired bus.

...




4. Select the PROFINET device for which you want to display details.




5. Press the "Details" softkey.
   The "PROFINET Diagnostics ... - Details" window opens.

### See also

Accessible nodes (Page 43)

## 10.4 Trace

The trace provides you with an oscilloscope function that allows the following data (signals) to be graphically displayed:

- General NC data
- PLC data

#### Trace session

The function serves for troubleshooting and resolving faults as well as to analyze the machine and process performance. In a session, data (signals) are recorded briefly before and after an event.

#### Graphic display

The recorded data is optically displayed in the form of curves, the so-called characteristics graphs.

You have the option of individually setting the recording type.

#### Saving and downloading data

You can store the recorded data in an XML file. The saved file can then be re-loaded so that it can be evaluated at a later time.

#### Recording fundamentals: Signals (data)

Signals (data) specify what is to be recorded:

- NC variables (axis data, drive data, channel data, etc.)
- PLC data

Cyclic events


Interpolation clock cycle (IPO clock cycle)

PLC cycle OB1


Position control clock cycle (servo clock cycle)

Non-cyclic events



#### Starting a session

- Recording using a trigger
- Manually starting and stopping a trace

## Analyzing the result

The following functions are available to analyze the result:

- Scaling characteristic graphs
- Zooming characteristic graphs
- Defining a characteristic range using markers

## 10.4.1 General procedure

**Sequence**

- Create a trace session file as follows:
  - Select the variable and define its representation type.
  - Set the graphics window, and the save mode
  - Set the trigger property to define the start and end of recording
- Save the recording and its associated properties that have been set
- Analyze the result by making the settings for the display in order to view additional details (e.g. zoom in, scroll, select section, etc.)

### See also

Variable for Trace (Page 329)

Trace settings (Page 334)

Trace options (Page 336)

Saving the trace file (Page 327)

Evaluate a trace (Page 337)

## 10.4.2 Trace session

### 10.4.2.1 Creating a session file

**Procedure**




1. Select the "Diagnostics" operating area.



2. Press the menu forward key.




3. Press the "Trace" softkey.







4. Press the "Load trace" and "New" softkeys.
   The "New Session" window opens in which the "Session" name is recommended. The file format is automatically specified as ".xml".

5. Supplement the file name or enter a new file name.




6. Select a variable type and press the "OK" softkey.
   You return to the "Select Variables for Trace: ...." ...." window.
   The name of the session that has been newly created is displayed in the header.

## See also

Variables filter/search (Page 329)

### 10.4.2.2 Saving the trace file

There are two options of saving a trace session:

- Variables and settings: Only the variables and the settings are saved.
- Variables, settings and recorded values: The variables, settings and the recorded values are saved.

## Procedure




1. You have edited an active trace session.




2. Press the "Store trace" softkey.
   The "Save Trace: Session ... " window opens and offers the following storage options.




3. Activate the corresponding checkbox and press the "OK" softkey.
   The "Store Trace" window opens and specifies the file name in the "Name of trace file:" entry field. The file format is automatically specified as ".xml" and cannot be changed.




4. Press the "OK" softkey to save the session file.
   If the session file already exists, then you obtain a corresponding confirmation prompt.
5. Press the "OK" softkey if you wish to overwrite the existing session files.
   - OR -




   Press the "Save as" softkey if you do not wish to overwrite the existing session file.
   A new name is suggested but you can specify a name yourself.
   - OR -




   Press the "Change directory" softkey if you wish to store the session file in a directory other than in the standard "Trace" directory.
   The data tree opens and you can select between the configured drives.




   Then press the "New directory" softkey if you wish to create a new directory.
   The "New Directory" window opens in which you can assign a directory name.




Press the "OK" softkey.

The "Store Trace" window opens in which the names of the target directory and the session file are displayed.

You can change the name of the session file.




6. To save the session file, press the "OK" softkey one more time.

- OR -

If the session file with this name already exists, then you obtain a corresponding confirmation prompt.

### 10.4.2.3 Load trace session file

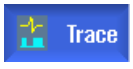
You have the option of loading already saved session data, to restart them or to display the trace that has been recorded.

| Settings | Meaning |
|---|---|
| Variables, settings | You can restart a trace session that has already been created. |
| Variables, settings and recorded values | You can view the result of a trace session, change the variable and if required, restart. |

## Precondition

A session file of recorded traces is available.

## Procedure




1. You have selected the "Trace" function.




2. Press the "Load trace" softkey.

The "Load Session File: Please Select File" window opens.




3. Select the desired session file from the appropriate directory, e.g. "Trace", and press the "Display trace" softkey.

The "Load trace: session..." is opened.

4. Select the checkbox "Variables, settings", if you wish to immediately start the trace session.




5. Press the "OK" softkey and the "Start trace" softkey to restart the trace.




- OR -

Select the checkbox "Variables, settings and recorded values", if you wish to display the trace session.




Press the "OK" softkey.

The data of the selected session are loaded displayed in the "Trace" window.

## 10.4.3 Variable for Trace

### 10.4.3.1 Variables filter/search

#### Selection of variables

To limit the selection of variables, set a filter and/or enter an initial value for the search. The following filters are available for selection:

| Filter | |
|---|---|
| Drive parameters | Displaying the drive parameters |
| System variables | Displays the system variables |
| NC | Display of NC, axis and channel signals as well as signals of the mode group in the OPI notation. |
| Axis | |
| Channel | |
| Mode groups | |
| Servo | Displays the servo-trace signals |
| PLC | Displays the PLC signals |
| PLC alarms | Displays the manufacturer-specific PLC alarm signals |
| All (no filter) | Displays all signals without filter: NC, axis and channel setting data as well as signals of the mode group in the OPI notation. |

**Note**

**"Filter/Search" when inserting variables**

The start value for "Filter/Search" of variables differs.

For example, to insert the variable $R[0], set "Filter/Search":

- The start value is 0, if you filter according to "System variables".
- The start value is 1, if you filter according to "All (no filter)". In this case, all signals are displayed and shown in the OPI notation.

#### Procedure

For a trace session, in the window "Selected Variables for Trace: Session ... " set the variables whose signals are to be traced.




1. In the "Select variables for trace: Session..." window, a trace session is displayed.
2. Position the cursor in the "Variable" column in an empty line.

   - OR -

   Position the cursor in the "Variable" column on a variable that you wish to replace by another.




3. Press the "Insert variable" softkey.

   The "Select Variable" window opens.

**Note:**

When selecting a "Drive parameter", firstly select the drive object (DO) and then insert the desired parameter from the list of corresponding parameters.

Filter/ search

4. Press the "Filter/search" softkey to start a specific search.

   In the "Filter" selection box, select a variable area.

   - OR -

   Directly enter the variable into the "Search" entry field.

   The cursor jumps to the term that is found.

Clear search

Press the "Delete search" softkey if you wish to terminate the search.

Add

5. Press the "Add" softkey to enter the required variable into a trace session.

   The "Variable Attributes" window opens and you can select the appropriate value (e.g. channel data, drive bus no.) from a combobox.

   - OR -

Replace

Press the "Replace" softkey to replace an already selected version by another one.

OK

6. Press the "OK" softkey to add or replace the variable.

   Each variable is automatically allocated a reference number, which is displayed in the upper section of the window. This number is opened automatically allocated. If a version is removed, then the following variables are re-numbered.

7. Now, select the desired settings for display, such as color, pen, qty. event etc.

### Editing variables of a trace session

| Function | Meaning |
|---|---|
| Ctrl + X<br>or right-click, shortcut menu "Cut" | Press the key combination or open the shortcut menu in order to cut out a variable. |
| Ctrl + C<br>or right-click, shortcut menu "Copy" | Press the key combination or open the shortcut menu in order to copy a variable. |
| Ctrl + V<br>or right-click, shortcut menu "Paste" | Press the key combination or open the shortcut menu in order to insert a variable. |

## 10.4.3.2 Selecting attributes of a variable

### Attributes

For a trace session, select the variables and assign the corresponding attributes:

| Column | Meaning |
|---|---|
| Variable | Address of the variables |

| Attributes | Meaning |
|---|---|
| Comment | • Name of the selected variables<br>• A general description for the variables. |
| Color | Color selection, the representation of the characteristic graphs.<br>When creating a variable, the next color from the available palette is automatically used. |
| Pen | Line type selection to display the characteristic graphs.<br>• No line<br>• Solid line<br>• Dashed line<br>• Dash-dot line<br>• Dash-dot-dot line<br>• Dotted line<br>• Stepped line XY<br>• Stepped line YX<br>• "* line" (line made up of * symbols)<br>• + line<br>• X line |
| Qty. | Defines whether the characteristic of the variables is displayed in the "Trace" window. If the checkbox is not selected, then no characteristic graphs are displayed. |
| Events | Event selection which triggers that the signal is traced: |
| | • Interpolation clock cycle (IPO clock cycle)<br>• Interpolation cycle (IPO2)<br>• Position control clock cycle (servo clock cycle)<br>• PLC cycle OB1<br>• Drive trace rate<br>• Start geometry axis/direction change<br>• Stop geometry axis<br>• Machine axis start / direction change<br>• Machine axis stop<br>• NC start (program runs)<br>• NC start (program ran)<br>• Start of the data recording |

| Attributes | Meaning |
|---|---|
| | • Block start, type 1 or block end (all program levels, without intermediate blocks)<br>• Block start, type 2 or block end (all program levels, with intermediate blocks)<br>• Block start, type 3 or block end (all main programs, without intermediate blocks)<br>• Block start or block end, block search (all program levels, without intermediate blocks)<br>• Geo axis start / direction change (2nd event)<br>• Geo axis stop (2nd event)<br>• Block start, type 2 (2nd event: all program levels, without intermediate blocks)<br>• Block end, type 2 (2nd event: all program levels, with intermediate blocks)<br>• Block start, block advance, type 1 (2nd event)<br>• OEM test event 1 (non-cyclic, block advance)<br>• OEM test event 2 (non-cyclic, main run) |
| | • Activate/deactivate synchronized action<br>• Synchronized action initiated (condition fulfilled)<br>• Signaled alarm<br>• Cancel softkey pressed<br>• Program level change<br>• Block end (Interpreter)<br>• Trace end (last event!)<br>• Start trace |
| | • WRTPR part program command<br>• WRTPR part program command (block search)<br>• Tool change<br>• Cutting edge change<br>• Tool change (block search)<br>• Cutting edge (block search)<br>• Start trigger initiated<br>• Stop trigger initiated<br>• Block end (block advance)<br>• Block end (2nd event: Block search)<br>• WRTPR part program command (block advance)<br>• Tool change (block advance)<br>• Cutting edge (block advance) |
| Event channel | This means that the channel can be defined in SINUMERIK in which a specific event occurs. For events which are not channel-specific, the field remains empty. |

| Attributes | Meaning |
|---|---|
| Bit mask | In the case of an integer variable, a bit mask can be specified in this field. For a bit mask, the particular signal value is AND'ed before being displayed. After the mask has been applied, all of the selected bits are moved to the right so that it looks as if the bit or the bits all start with bit zero.<br>This means, if all bits with the exception of bit 7 have been masked, then the integer number that would have been obtained would either have a value of 0 or 1, however, not 0 or 128.<br>If all of the bits with the exception of bit 7 and 0 have been masked, the resulting integer number would either have the value 0, 1, 2 or 3, however, not 0, 1, 128 or 129. |
| Decimal places | This setting is used to define how many places to the right of the decimal point are displayed at the axis identifiers. |
| Coord. axis | The coordinate axes are displayed to the left or right in the graphic window or there is no display. |
| Display Y | Value input or 0 |
| Scale factor | Defines the scale. |
| Units | Displays the measurement unit, e.g. mm/min. The system specifies this and it cannot be changed. |

### 10.4.3.3 Displaying details of a variable

You can display the most important information and settings of the display at a glance in the detail window, i.e. you do not have to use the scroll bar for example to get to the "Event" column.

**Displayed information**

- Variable address
- Comment with a description of the variables
- Events
- Channel, axis, access level, mode group, etc.
- Smaller graphic display with the event; when this event occurs the characteristic graphs are recorded as well as their settings, such as color, line type, etc.

## Procedure




1. You are in the "Select Variables for Trace: ..." window. ...".
2. Position the cursor to the required variable.




3. Press the "Details" softkey.

   A subscreen with the most important information as well as the graphic display is shown in the lower half of the window.




   Press the "Details" softkey again to display the subscreen.

## 10.4.4 Trace settings

### 10.4.4.1 Trace settings (PLC, NC, servo)

In the "Trace Settings (PLC, NC, Servo)" window, you define the conditions for starting and stopping the trace session.

| **Condition for Start trace** | |
|---|---|
| With "Start trace" softkey | The recording is started immediately after pressing the "Start trace" softkey. |
| If variable | The session is activated by a variable. The desired variable is selected via the "Insert variable" or "Quick list" softkey. |
| • Condition and value | After the session has been activated, this defines at which value the trace recording is started.<br>The following values are offered: = =, ≠, >, >=, <, <=, ascending, descending, changed. |
| **Data acquisition [s]** | |
| Specifies where the recorded data is saved:<br>• With "On NCK", the recorded data is saved in the NC work memory until the recording is stopped. Only then is the session file saved on the CompactFlash card.<br>• With "CF card" or "On hard disk", the data is saved during the recording on the CompactFlash card or hard disk (e.g. of the PCU). As a consequence, more signals can be processed. | |
| **Condition for Stop trace** | |
| With "Stop trace" softkey | The recording is stopped after pressing the "Stop trace" softkey. |
| If the time has expired | Defines the period which after expiry, the trace recording is exited. |
| If the memory is full | The trace session is terminated when the specified data volume (KB) is reached. |
| If variable | The recording is stopped by a variable. The desired variable is selected via the "Insert variable" or "Quick list" softkey. |
| • Condition and value | After the session has been activated, this defines at which value the trace recording is exited.<br>The following values are offered: = =, ≠, >, >=, <, <=, ascending, descending, changed. |
| • Delay time [s] | This defines how long the trace still runs after the session has been exited. |
| • Overwrite at memory limit (KB) | Defines from which memory limit the data is overwritten (ring buffer). |

### 10.4.4.2 Trace settings (drive)

In the "Trace Settings (Drive)" window, you define the conditions for starting and stopping the trace session:

| **Condition for Start trace** | |
|---|---|
| With "Start trace" softkey | The recording is started immediately after pressing the "Start trace" softkey. |

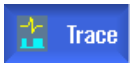
<table>
<tr><td>If variable</td><td>The recording is activated after pressing the "Start trace" softkey when the selected variable clearly satisfies the condition and the specified value.<br>The following conditions are offered:<br>• Positive edge<br>• Negative edge<br>• With bit mask<br>The "Bit mask" softkey is displayed when the "Bit mask" condition is selected in order to select the desired bits. Irrespective of the maximum value (hexadecimal) which is possible through the bit mask, a "Value" (decimal) can be entered at which the trace recording starts precisely.<br>Example:<br>The trace recording is started when the following bits are set: Bit 0 = 0; bit 1 = 1; bit 2 = 0; bit 3 = 1
<table>
<tr><th>Bit mask = fH</th><th>Condition</th><th>Value (decimal)</th></tr>
<tr><td>Variable: CU/ r722/3.3.1</td><td>With bit mask<br>☑ Bit 0<br>☑ Bit 1<br>☑ Bit 2<br>☑ Bit 3</td><td>10</td></tr>
</table>
</td></tr>
<tr><td>Preprocessing time [s]:</td><td>The input of a preprocessing time corresponds to the functional parameter p4722 "Trace trigger delay":<br>• A positive preprocessing time starts the recording before the trigger event.<br>• A negative preprocessing time starts the recording after the trigger event.<br>When entering a preprocessing time, the entire duration of the recording must be taken into account:<br>• If the preprocessing time is greater than the entire duration of the recording, the entry is rejected.<br>• If too large a preprocessing time is entered, the trigger event may not be recorded.</td></tr>
<tr><td colspan="2">Data storage on CU</td></tr>
<tr><td>Drive trace rate [µs]:</td><td>Enter a factor with the "Sampling rates" softkey to set the effective sampling rate. With a higher sampling rate, the recording duration is also longer.</td></tr>
<tr><td colspan="2">Condition for Stop trace</td></tr>
<tr><td>If the time has expired</td><td>Defines the period which after expiry, the recording is exited.</td></tr>
<tr><td>If the memory is full</td><td>The recording is terminated when the specified data volume (KB) is reached.</td></tr>
</table>

### 10.4.4.3 Trace options

You define the following properties of a trace session in the "Trace - Options" window.

- Setting the graphic window
  - Displaying main grid lines: You define whether a horizontal grid is displayed and in which color.
  - Displaying grid sub-lines: You define whether a vertical grid should be displayed and in which color.
  - Background: You select the background color of the graphic window.
- Defining storage options
  - Activate the "Confirm file overwrite" checkbox in order to receive a confirmation prompt each time that a trace session is to be overwritten. If the checkbox is deactivated, the trace session is overwritten without a confirmation prompt.
  - Activate the "Export data in csv file" checkbox in order to also save the data as a csv file in addition to the xml session file. This file only serves to export the data and cannot be read in again.

#### Procedure




1. You are in the "Select Variables for Trace: ..." window. ..." and a trace session has been selected.




2. Press the "Options" softkey.
   The "Trace - Options" window opens.




3. Select the desired settings and press the "OK" softkey in order to accept the settings.

### 10.4.4.4 Starting the trace

In the "Trace" window, you obtain the curve with the data of the variables selected for a trace session.

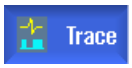
The diagram comprises two Cartesian coordinates:

- Time axis
  The horizontal x axis supplies the time values.
- Value axis
  The vertical y axis displays the resolution fineness (amplitude).

#### Recording the trace session

- Manually starting and stopping a trace
- Starting and stopping recording using a trigger

### Procedure




1. You have listed the desired variables for a trace section.




2. Press the "Display trace" softkey.
   The "Trace" window is opened. The recording is loaded and the procedure is displayed in progress window.




3. Press the "Start trace" softkey if you wish to immediately start recording.

   - OR -

   The recording is triggered by the trigger condition.




4. If you wish to manually end the recording, press the "Stop trace" softkey.

   - OR -

   The recording is stopped by the trigger condition.

## 10.4.5 Evaluate a trace

### 10.4.5.1 Setting trace views

In order to permit a precise evaluation of the recorded signals, you have the option to edit the view as well as the display of the characteristic graphs:

- Selection/legend
  - Show legend: In addition to the graphic display, the variable data are displayed as comment, numbered and in color.
  - Make a selection: You define which variables are displayed in the graphic window.
- Scale
  - Specify scaling factors for minimum and maximum X and Y value
- Zoom
  - Increasing or decreasing the section
- Cursors: The cursor display provides functions that allow one, two or both curves to be precisely measured.
  - Cursor A
  - Cursor B
  - Both cursors
  - Snap to curve: The cursor precisely moves along the curve
  - Point mode: Cursor jumps directly to various values.
  - High point mode: The cursor jumps from one peak value to the next.
  - Low point mode: Cursor jumps to the lowest values.

### Precondition

The "Trace" graphic window is opened and the trace has been recorded.

### Editing curves




You jump from <TAB> one variable to the next using the <TAB> key.




You jump back using the <Ctrl> and <TAB> keys.









With the cursor keys, you select marks, move to individual values and can define a zoom range.

## 10.4.5.2 Selecting a variable

You have the option of displaying the selected variables as legend and to make a selection using a checkbox.

| Legend |
|---|
| Numbering the variables |
| Axis identifier |
| Comment of the variables |

### Displaying a legend and selecting a variable




1. A recording is opened in the "Trace" graphics window.




2. Press the "Legend / selection" softkey.
   The graphic window is halved. The variables are displayed as comment and with checkbox in the left-hand section of the window.
3. Activate or deactivate the required variable using the checkbox to display it as curve.




4. Press the softkey again to hide the legend area again.

### 10.4.5.3 Scaling the display

The scaling allows you to make a precise measurement of selected variables from the X time axis as well as from the Y value axis. You have the option of saving the larger or smaller display in a user-defined way

| Setting | Meaning | |
|---|---|---|
| X minimum or X maximum | Highest or lowest value of the X time axis. The values are used for recording when the trace is started again. | |
| | Fixed scale, X axis | Values of the X time axis are kept. |
| | Strip chart recorder mode | The values of the X time axis are used while recording. |
| Y minimum or Y maximum | Defining the highest or lowest value of the Y value axis (amplitude). | |
| Offset or resolution | Defining the rms values (mean values) of the amplitudes.<br>The scale is changed by a factor of 10. | |

## Scaling the time and value axis




1. A recording is opened in the "Trace" graphics window.




2. Press the "Scale" softkey in order to scale the display.




The "Select X-Y Scaling" input window opens and the values of the X time axis are displayed.




3. Press the "X time axis" softkey again if you wish to hide the values.

- OR -




Press the "Y selected curves" softkey to display the values of the Y value axis.




Press the softkey again if you wish to hide the values.

- OR -

You have selected both axes.




4. Press the "Scale +" or "Scale -" softkey until the desired scaling is reached.




- OR -

Enter the scaling values directly into the input window and select the appropriate property.




5. Press the "Scale" softkey to save the settings.

- OR -

Fit selected — Press the "Adapt selected" softkey to display the selected characteristic graphs in the full graphic window.

Fit one — Press the "Fit one" softkey to display the selected characteristic graphs in the full graphic window.

### 10.4.5.4 Zooming the display

Using the zoom function, you have the option of viewing selected sections of curves in detail. After you have defined an area or a section, you can increase or decrease the size of it (zoom in or zoom out).

## Procedure

Trace

1. A recording is opened in the "Trace" graphics window.

Zoom

2. Press the "Zoom" softkey if you wish to zoom-in or zoom-out on the characteristic graphs.

X - time axis

Y - value axis

3. The "X time axis" or "Y value axis" softkey is selected.

Fit separated

4. Press the "Adapt individually" or "Adapt all" softkey if you wish to display all selected characteristic graphs in a separate area positioned one above the other.

or

Fit all

- OR -

Fit overlaid

Press the "Adapt together" softkey if you wish to adapt all of the characteristic graphs to the graphic window.
As all of the curves fill the complete window area, they are displayed superimposed on one another.

- OR -

Fit X separated

Press the "Individually adapt X" softkey if you wish to normalize the X time axis only in the graphic window.

- OR -

Fit Y separated

Press the "Individually adapt Y" or "Adapt all" softkey if you only wish to scale the Y value axis in the graphic window.

or

Fit all

- OR -




Press the "Adapt Y together" softkey if you wish to adapt the value axis of all characteristic graphs to the graphic window.
As all curves fill the complete window area, they are displayed superimposed.







5. Press the "Zoom +" or "Zoom -" softkey until the desired display size is reached.

**Zoom in/zoom out area**




Press the "Zoom area" softkey to define a specific section.

A frame is displayed.







Press the "Zoom +" or "Zoom -" softkey to change the frame step-by-step.




Press the "OK" softkey to display the zoom area.

### 10.4.5.5 Position the cursor

Using the cursor, you have the option of determining the precise values from a defined section. To do this, the cursor is positioned directly on the curve.

## Procedure




1. A recording is opened in the "Trace" graphics window.




2. Press the "Cursor A" softkey.
The reference number, color of the variables and the color of the coordinate crossing point are displayed on the softkey.





3. Position "(A)" at the position on the curve where you wish to start the measurement.
The coordinates are specified in a tooltip and in the status line.




4. Press the "Cursor B" softkey.
The reference number, color of the variables as well as the color of the coordinate crossing point are also displayed on this softkey.



Position "(B)" at the position on the curve where you wish to end the measurement.






5. Press the "Both cursors" softkey to display the area between A and B.
   The coordinates of A and B as well as the difference "Δ" are displayed in the status line.







Press the active "Cursor A" and "Cursor B" softkeys again to deselect the position.
When re-selected, the positions are reset.




Press the "Crop to screen" softkey to display the area, defined by "Cursor A" and "Cursor B", in the complete graphic window.



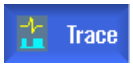

Press the "Snap to waveform" softkey to move the cursor along the curve step-by-step.

### 10.4.5.6 Acquiring measured values

You have the option of directly determining the following values of a curve:

- Peak-to-peak value
- Peak-to-valley value

## Procedure




A recording is opened in the "Trace" graphics window.




Press the "Point Mode" softkey to continuously move the cursor along the curve.




Press the "Peak Mode" softkey to position the cursor at the highest value.




Press the "Valley Mode" softkey to position the cursor at the lowest value.




If you press the "Peaks" softkey, the cursor only goes to the peak-peak values.




If you press the "Minimum" softkey, the cursor only goes to the peak-valley values.

# 10.5 Drive system

## 10.5.1 Displaying drive states

### Status display

The "Drive System Diagnostics" window displays the status information about the drive units and the associated drive objects. The status symbols have the following meaning:

| Display icon | | Meaning |
|---|---|---|
| | Green | The drive object is in (cyclic) operation, without any detected problems. |
| | Yellow | The drive object has detected a less serious problem, i.e. a warning is pending, for example, or enables are missing. |
| | Red | This drive object has detected a serious problem, for example, an alarm is pending. |
| | Gray | The drive status could not be determined for this drive object. |
| # | Special characters | Error when reading the data. |

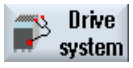
### Procedure




1. Select the "Diagnostics" operating area.



2. Press the menu forward key and the "Drive system" softkey.













3. The "Drive System Diagnostics" window opens.
   The name of the selected drive object is displayed in the window title.
   Press the "Drive object +" or "Drive object -" softkey.
   The next (+) or previous (-) drive object is selected.

   - OR -

   Press the "Select drive unit" softkey.
   The "Select Drive Unit" window opens.
4. Select the desired drive unit via the drop-down list box and press "OK" to confirm the selection.

### See also

Displaying details of the drive objects (Page 344)

## 10.5.2 Displaying details of the drive objects

### Drive system diagnostics - details

The following information is displayed for the drive objects:

- Status display (r0002)
  - The status display (parameter r0002) of the drive object is displayed.
- Missing enable signals (r0046)
  - The enable signals are diagnosed except for the drive objects that do not have enable signals (e.g. Control Unit). Missing enables are shown in a pane below the displays.
- Commissioning parameter filter
  - (p0009) – control unit
    The value of the "Device commissioning parameter filter" control unit parameter (p0009) is displayed.
  - (p0010) – further drive object
    The value of the "Drive object commissioning parameter filter" parameter (p0010) is displayed.
- Current fault
  The alarm number of the actual drive object fault is displayed or "No fault present", when no drive object fault is pending.
  You obtain an overview of the pending faults with the following data in an additional window:
  - Alarm number
  - Time: Date and time
  - Alarm text
- Current warning
  The number of the current drive object warning is displayed or "No warning present", when no drive object warning is pending.
  You obtain an overview of the pending alarms with the following data in an additional window:
  - Alarm number
  - Time: Date and time
  - Alarm text

### Meaning of the symbols

| Display icon | | Meaning |
|---|---|---|
| | Green | The relevant value of the drive object signals no problem detected. |
| | Yellow | The relevant value of the drive object signals a less serious problem, i.e. a warning is pending, for example, or enables are missing. |
| | Red | The relevant value of the drive object signals a a serious problem, for example, an alarm is pending. |

| Display icon | | Meaning |
|---|---|---|
| ○ | Gray | The drive status could not be determined for this drive object. |
| # | Special characters | Error when reading the data. |

### Procedure




1. Select the "Diagnostics" operating area.



2. Press the menu forward key and the "Drive system" softkey.




The "Drive System Diagnostics" window opens.

3. Select the drive object for which you want to display further details.




4. Press the "Details" softkey.

The "Drive System Diagnostics - Details " window opens.

The name of the drive object is displayed in the window header.




Press the "Faults" softkey.

The "Drive Unit Faults" window opens that shows an overview of the faults that have been output.




Press the "Alarms" softkey.

The "Drive Unit Alarms" window opens that shows an overview of the alarms that have been output.




You can display the pending SI messages if Safety Integrated is integrated in your controller.

## 10.6 Action log

### 10.6.1 Setting the action log

In the "Action Log Settings" window you specify whether the operating sequences are to be logged.

Here, you select which control actions should be logged, so that operator actions and sequences can be tracked and understood at a later date.

When the action log is active, you can display recorded traces in the system data using the following files:

- "actual_actionlog.com", actual action log report
- "actual_crashlog.com", crashlog backup

## Data that can be logged

| | |
|---|---|
| Logging on | Logging is activated or deactivated. |
| Alarm status change | Incoming and outgoing alarms are logged. |
| Keyboard actions | All actions on the operator panel front and on an external keyboard are logged. |
| Channel status change | NC/PLC states are logged via the information channel status. If this information can be obtained in a timely manner, in many cases these states can be used to verify the operation of the machine control panel. |
| Window switchover | The form and dialog names (assigned by the programmer) are logged each time a window is opened or closed. |
| Writing NC/PLC data | The writing of NC and PLC variables is logged. |
| File access | Copying to the NC is logged. |
| Function calls in the NC (PI service) | Some program calls, for example, ASUB, are logged. |
| Curr. program status | Additional information for particular events is activated. For important alarms that require NC stop, NC start, or NC reset, actual values and the current block are recorded for example.<br>The program ascertains which channel and which spindle are used for the additional information. |
| Write interval file | Settings for updating files:<br>• "automatic": The action log saves the information to an internal buffer. If the buffer is full the entries are written to the CF card. The entries may be lost when the control is switched off.<br>• "after every event": All entries are backed up immediately, this prevents data loss due to a power failure or similar.<br>Notice: As CF cards only permit a limited number of write accesses, this setting is not recommended for normal operation.<br>• "time-controlled": New entries are backed-up for a particular length of time. An additional input field appears in which you can specify a time in seconds. |
| Save log as file for alarm(s) | The alarm numbers for which a "crash log" is generated are specified. The alarms are entered separated by a comma. |

## Generating a crashlog file

The "actual_crashlog.com" is a backup file of the actual log when a critical event occurs (for example, emergency stop alarm).

The file is then no longer stored in the ring buffer of the action log and can no longer be overwritten with new entries.

The entries in the crash log file are only overwritten when yet another critical events occurs.

At the beginning, the file has no entries and is only written to after the following event:

- The alarm entered in the "Save log as file for alarm(s)" entry field is output.

### Procedure




1. Select the "Start-up" operating area.










2. Press the "HMI", "Diagnostics", and "Action log" softkeys
   The "Action Log Settings" window opens.
3. Activate the "Logging on" checkbox to activate the action log.
4. Activate the relevant checkbox if you want to acquire particular events in the action log.

### See also

Displaying the log file (Page 347)

## 10.6.2 Displaying the log file

Loggable data that was logged when the action log function was active is displayed in the "Action Log File" window.

### Log files

The actual data can be displayed using the following files:

- "actual_actionlog.com"
- "actual_crashlog.com"

### Procedure




1. Select the "Start-up" operating area.




2. Press the "System data" softkey.
   The data tree is displayed.
   You will find the saved files in the HMI data folder under /logs/action log.

3. Select the desired com file and press the "Open" softkey or double-click the file.




4. Press the "Display new" softkey to update the display of events in the log (actual_actionlog.com).

   The entries made since the log file was called up are displayed.

## 10.6.3 Searching in the log files

You can search for specific events in the log.

### Precondition

The relevant log file is open.

### Procedure




1. Press the "Search" softkey.

   A new vertical softkey bar appears. The "Search" window opens at the same time.

2. Enter the desired search term in the "Text" field.




3. Position the cursor in the "Direction" field and choose the search direction (forward, backward) with the <SELECT> key.




4. Press the "OK" softkey to start the search.

   If the text you are searching for is found, the corresponding line is highlighted.




Press the "Continue search" softkey to continue the search, as necessary.

- OR -




Press the "Cancel" softkey when you want to cancel the search.

### Additional search option




1. Press the "Go to End" softkey to scroll to the end of a large log. You get to the oldest entry of the recording.




2. Press the "Go to Beginning" software to return quickly to the most recent log entry.

## 10.6.4 Storing a log

You can store the displayed log in the "HMI data/Logs/Alarm logs" directory. It is stored as a binary file as well as ASCII file. You can read an ASCII file with any editor.

The following logs are generated:

- action.com (binary file)
- action.log (ASCII file)
- crash.com (binary file)
- crash.log (ASCII file)

### Requirement

The relevant log file is open.

### Procedure




1. Press the "Save log" softkey.
   The log is stored in the "HMI data/Logs/Alarm logs" directory.

**Note**

The "Save log" softkey is only available for files that have not yet been saved.

## 10.6.5 Structure of a log file

### Content of the log file

The following data is recorded for every logged event:

| | | | |
|---|---|---|---|
| HMI version | | NC version | |
| System | Entry keyword | | Date/time |
| | Entry text | | |
| | DETAILS | | |

| Entry | | Description |
|---|---|---|
| HMI and NC version output | | |
| System: | | |
| | HMI | Keystrokes, window change |
| | NC | Write variables, PI services |
| | MSG | Alarm entries |
| | USR | Entries generated by the machine manufacturer for the user |
| | ERR | Action log error, e.g. log file not found |
| Entry keyword: | | |
| | HMI_START | Entry of an HMI boot |
| | HMI_EXIT | Entry of an HMI shutdown |

| Entry | | Description | |
|---|---|---|---|
| | PLC_CRASH | Entry of a PLC crash | |
| | KEY_PRESSED | Entry of a keystroke | |
| | KEY_HOLD | Entry of a key hold | |
| | KEY_RELEASED | Entry of a key released | |
| | ALARM_ON | Entry of an incoming alarm event | See below "Alarm display" |
| | ALARM_OFF | Entry of an alarm going event | |
| | ALARM_ACK | Entry of an alarm acknowledgement event | |
| | OPEN_WINDOW | Entry of a window opening | |
| | CHN_STATE_CHANGED | Entry of a channel status change | |
| | OPMODE_CHANGED | Entry of an operating mode change | |
| | TOOL_CHANGED | Entry of a tool change | |
| | OVERRIDE_CHANGED | Entry of override changes | |
| | DOM_CMD | Entry of a download to the NC | |
| | PI_CMD | Entry of a PI service | |
| | WRITE_VAR | Entry of a write to NCK/PLC variable | |
| | AREA_CHANGED | Entry of an area changeover | |
| | NC_CONNECTION | Entry of an coming/going NC connection | |
| | USER | User entry via the OEM interface | |
| | ACTIVATED | Action log was activated | |
| | DEACTIVATED | Action log was deactivated | |
| | INTERNAL | Internal action log entry | |
| Date/time | | Date and time of the event<br>For HMI_Start, PLC_Crash or date change since the last entry, the intermediate period between the entry keyword and date is designated with an "-". | |
| Entry text | | Event data is written in plain text. | |
| DETAILS | | If additional information regarding an entry is known, e.g. a recorded IPO trace. | |

## Example

**Note**

**DB19.DBW24 interface signal**

The DB19.DBW24 interface signal used in the example does not apply to the SINUMERIK 840D sl. It is DB1900.DBW4 for the SINUMERIK 828D.

```
NCK    WRITE_VAR                                    18.02.2009  09:40:12
       ncu1.local wrote variable: DB19.DBW24 = 0
HMI    KEY_PRESSED                                  18.02.2009  09:40:12
       Key up: "Return" (1000004/ 0)
HMI    HMI_START ----------------------------       18.02.2009  09:34:15
       HMI started.
```

### Alarm display

```
MSG     ALARM                                                Date/time
        No.       Deletion criterion   Text
        DETAILS
```

| Entry | | Description |
|---|---|---|
| MSG | | |
| ALARM | | Type of alarm: ALARM_ON, ALARM_OFF, ALARM_ACK |
| Date/time | | Data and time of the event |
| No. | | Alarm number |
| Deletion criterion: | | Information as to how the alarm was acknowledged |
| | AUTOMATIC | Automatic acknowledgement |
| | POWER ON | Acknowledgement using PowerOn |
| | RESET | Acknowledgement using NCK reset |
| | NC START | Acknowledgement using NC start |
| | NC RESET | Acknowledgement using NC reset |
| | ALARM CANCEL | Acknowledgement using Alarm Cancel key |
| | RECALL | Acknowledgement using the Recall key |
| | HMI | Acknowledgement using HMI |
| | PLC | Acknowledgement using PLC |
| Text | | Display of the English alarm text |
| DETAILS | | If additional information regarding an entry is known. |

### Example

```
MSG    ALARM_ON                                    20.02.2009   14:25:37
       8020   POWER ON   : Option 'activation of more than 1 channels' not
       set
       DETAILS:
       Mode: JOG Program: canceled Channel: interrupted
       Program-Level information:
       Level Program running:                     Invoc        Offset
       1      /_N_MPF0                            0            0
```

## 10.6.6 Advanced settings

You define the size of the log file of the action log in the "actlog.ini" configuration file.

### Procedure

1. You can copy a sample configuration file "actlog.ini" from the following directory: /siemens/sinumerik/hmi/template/cfg.
2. Save or create the file in the /oem/sinumerik/hmi/cfg or /user/sinumerik/hmi/cfg directory.
3. Open the file and after "; ActionLogSize=", enter the file size (in bytes).

### "actlog.ini" configuration file

```
Template for the configuration of Action Log
;
; To activate the settings remove the
; commentary ';' at the beginning of the line
[ActionLogSettings]
;To change the path of the internal action log file use the following setting
;ActionLogPath=/user/sinumerik/hmi/ac_log/action.com
;To change the size of the internal action log file use the following setting
;ActionLogSize=5000000
;To change the path of the internal crash log file use the following setting
;CrashLogPath=/user/sinumerik/hmi/ac_log/crash.com
```

## 10.7 HMI trace

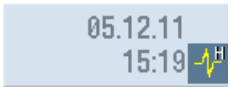
### Creating an HMI trace

The "HMI trace" function is available for extended diagnostics of the operating software. The prerequisite is at least access level 3 (user).

Procedure to create an HMI trace file:

1. An expert of the Service & Support Center provides a sltrc_keybitset.ini file. Copy the sltrc_keybitset.ini file to a USB memory medium and load the file on the controller or operator panel.
2. In the operating area, select "Start-up" → "HMI" → "Diagnostics >" → "HMI trace".
3. Select the INI file on the USB memory medium with the "Load configuration" softkey.

4. Start the trace by activating the checkbox or from any place on the user interface with the following key combination:
   <Alt> + <T> on the operator panel
   <Alt> + <Shift> + <T> on an external keyboard
   The following icon is displayed at the top right in the header to show that a trace is active:

5. On the user interface, repeat the operating sequence whose cause is to be clarified.
6. Stop the trace by deactivating the checkbox or from any place on the user interface with the following key combination:
   <Ctrl> + <T> on the operator panel
   <Ctrl> + <Shift> + <T> on an external keyboard
7. A sltrc.out output file is generated which you can copy to the storage medium with the "Export data" softkey.
8. Send the sltrc.out file for analysis back to the Service & Support Center.

**Note**

On completion of the analysis, it is recommended to delete all files under the following path: ../user/sinumerik/hmi/log/sltrc

**Note**

**"Extended" softkey**

The other options in this dialog support suitably qualified service engineers in the analysis of the system and evaluation of the HMI trace.

# Appendix A

## A.1 Information about third-party software used

Copyright 1995 Sun Microsystems, Inc.

Printed in the United States of America.

All Rights Reserved.

This software product (LICENSED PRODUCT), implementing the Object Management Group's "Internet Inter-ORB Protocol", is protected by copyright and is distributed under the following license restricting its use. Portions of LICENSED PRODUCT may be protected by one or more U.S. or foreign patents, or pending applications.

LICENSED PRODUCT is made available for your use provided that you include this license and copyright notice on all media and documentation and the software program in which this product is incorporated in whole or part. You may copy, modify, distribute, or sublicense the LICENCED PRODUCT without charge as part of a product or software program developed by you, so long as you preserve the functionality of interoperating with the Object Management Group's "Internet Inter-ORB Protocol" version one. However, any uses other than the foregoing uses shall require the express written consent of Sun Microsystems, Inc.

The names of Sun Microsystems, Inc. and any of its subsidiaries or affiliates may not be used in advertising or publicity pertaining to distribution of the LICENSED PRODUCT as permitted herein. This license is effective until terminated by Sun for failure to comply with this license. Upon termination, you shall destroy or return all code and documentation for the LICENSED PRODUCT.

LICENSED PRODUCT IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND INCLUDING THE WARRANTIES OF DESIGN, MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE.

LICENSED PRODUCT IS PROVIDED WITH NO SUPPORT AND WITHOUT ANY OBLIGATION ON THE PART OF SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES TO ASSIST IN ITS USE, CORRECTION, MODIFICATION OR ENHANCEMENT. SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES SHALL HAVE NO LIABILITY WITH RESPECT TO THE INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY LICENSED PRODUCT OR ANY PART THEREOF.

IN NO EVENT WILL SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES BE LIABLE FOR ANY LOST REVENUE OR PROFITS OR OTHER SPECIAL, INDIRECT AND CONSEQUENTIAL DAMAGES, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

SunOS, SunSoft, Sun, Solaris, Sun Microsystems and the Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc.

SunSoft, Inc.
2550 Garcia Avenue
Mountain View, California 94043

Copyright (c) 1991 by AT&T.

Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software.
THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN PARTICULAR, NEITHER THE AUTHOR NOR AT&T MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

This product includes software developed by the University of California, Berkeley and its contributors.
QLocale's data is based on Common Locale Data Repository v1.6.1.

**Note**

You can find additional information on the product DVD in the Readme_OSS file about the third-party software used.

## A.2 Abbreviations

| Abbreviation | Meaning |
|---|---|
| ASCII | American Standard Code for Information Interchange:<br>American coding standard for the exchange of information |
| BAG | Mode group |
| OPI | Operator Panel Interface |
| CNC | Computerized Numerical Control: Computerized numerical control |
| CEC | Cross Error Compensation: Sag compensation |
| CP | Communication Processor |
| DCP | Discovery and basic Configuration Protocol: ProfiNet protocol |
| DHCP | Dynamic Host Configuration Protocol |
| DIN | Deutsche Industrie Norm |
| DNS | Domain Name System |
| DIR | Directory: Directory |
| DRAM | Dynamic Random Access Memory |
| EES | Execution from External Storage |
| ERTEC | Enhanced Real Time Ethernet Controller |
| ESR | Extended Stop and Retract |
| ETC | ETC key ">"; Softkey bar extension in the same menu |
| FRAME | Data block (FRAME) |
| FIFO | First In - First Out: Method of storing/retrieving data in/from a memory. |

| Abbreviation | Meaning |
|---|---|
| GDIR | Global Directory: Global part program memory |
| GUD | Global User Data: Global user data |
| MSD | Main Spindle Drive |
| HSC | High Speed Cutting: High-speed machining |
| INC | Increment: Incremental dimension |
| INI | Initializing Data: Initializing data |
| IP | Internet Protocol |
| IPO | Interpolator |
| ISO | International Standard Organization |
| JOG | Jogging: Setup mode |
| LED | Light Emitting Diode: Light emitting diode |
| LLDP | Link Layer Discovery Protocol |
| LUD | Local User Data: Local user data |
| MAC | Media Access Control |
| MB | Megabyte |
| MD | Machine data |
| MDA | Manual Data Automatic: Manual input |
| MCS | Machine Coordinate System |
| MLFB | Maschinenlesbare Fabrikatbezeichnung |
| MPF | Main Program File: NC part program (main program) |
| MCP | Machine Control Panel: Machine control panel |
| NC | Numerical Control: Numerical control |
| NCK | Numerical Control Kernel: NC kernel with block preparation, traversing range, etc. |
| NCU | Numerical Control Unit: NCK hardware unit |
| NFS | Network File System |
| NTP | Network Time Protocol |
| ZO | Zero offset |
| OEM | Original Equipment Manufacturer |
| OID | Object Identifier: Information identifier |
| OP | Operator Panel |
| PCU | Programmable Control Unit |
| PI service | Program Invocation Services |
| PG | Programming device |
| PLC | Programmable Logic Control |
| RFC | Request for Comments |
| REF | REFerence point approach function |
| REPOS | REPOSition function |
| ROV | Rapid Override: Input correction |
| RPA | R-Parameter Active: Memory area in NCK for R- NCK for R parameter numbers |
| SBL | Single Block: Single block |
| SD | Setting Data |
| SDB | System Data Block |
| SEA | Setting Data Active: Identifier (file type) for setting data |

| Abbreviation | Meaning |
|---|---|
| SK | Softkey |
| SMB | Server Message Block |
| SNMP | Simple Network Management Protocol |
| SPF | Sub Program File: Subroutines |
| SRAM | Static Random Access Memory: Static RAM (non-volatile) |
| SSH | Secure Shell: Network protocol |
| SW | Software |
| SYF | System Files: System files |
| TCU | Thin Client Unit |
| TEA | Testing Data Active: Identifier for machine data |
| TMA | Tool Magazine Active: Magazine data |
| TO | Tool Offset: Tool offset |
| TOA | Tool Offset Active: Identifier (file type) for tool offsets |
| UDP | User Datagram Protocol: Network protocol |
| UFR | User frame |
| VNC | Virtual Network Computing |
| FD | Feed Drive |
| WCS | Workpiece Coordinate System |
| TM | Tool Management |

# Index

## 3

3D simulation, 61, 65

## A

Access levels, 9
Access right
  Effect, 12
  Global, 11
  individual, 11
  Remote access, 11
Action log
  actlog.ini, 351
  Log file, 349
  Open log file, 347
  Saving log file, 349
  Search for events, 348
  Setting the log size, 351
  Settings, 345
action.com, 349
action.log, 349
actlog.ini, 351
Actual value window
  Setting font size, 47
actual_actionlog.com, 345
actual_crashlog.com, 345
Adapt tool measurement
  Examples when milling, 184
  for turning, 189
  when milling, 182
Adapt workpiece measurement, 178
Analog spindle, 197
Axis diagnostics, 309

## C

Channel operational message, 49
  Standard configuration, 50
Circular position pattern, 92
Circumferential groove, 92
Clamping
  Setting up function, 67
Configuring the tool change reason, 262
Contour grooving, 111
Contour milling, 91
Contour turning, 111
Coolant
  Assigning, 259
  Identifier, 271
Coordinate system
  MD52000, 72
crash.com, 349
crash.log, 349
CUST_800.SPF, 77
CUST_832.SPF, 77, 160
CUST_CLAMP, 77
CUST_M6, 77, 79
CUST_MEACYC
  Structogram, 165
CUST_MEACYC.SPF, 77
CUST_MULTICHAN, 77, 83
CUST_T, 77, 79
CUST_TECHCYC.SPF, 77
  Aligning, 81
Cutting edge parameters
  Identifiers, 230
  OEM identifier, 269
Cycle alarms, 134
CYCLE495, 128
CYCLE63, 91
CYCLE79, 92
CYCLE800
  Structogram, 153
CYCLE832, 159
CYCLE84, 76
CYCLE840, 76
CYCLE930, 111
CYCLE950, 111
CYCLE951, 111
CYCLE952, 111
CYCLE99, 110
CYCLE996, 157
Cycles version
  Define, 191
  Displaying, 191
Cylinder surface transformation
  Milling, examples, 96
  under milling, 95
  under turning, 115
  with groove side offset, 118
  without groove side correction, 117
Cylindrical grinding, 127
  Displaying softkeys, 71

## D

Date
Selecting the format, 16
Setting, 17
DHCP server detection, 42
Diagnostics
Ethernet/PROFINET network, 42
HMI trace, 352
Direction of rotation
Counterspindle, 107
Main spindle, 102
Drilling, 75
Drive
Execution from External Storage, 30
Global part program memory, 31
Logical drive, 29
Set EXTCALL, 30
Setting up, 29
Drive system
Diagnostics, 343
Diagnostics - details, 344

## E

Editing the tool and magazine management, 291
Ethernet/PROFINET
Diagnostics, 42
EXTCALL
switch off and on again, 29

## F

Face end machining
under turning, 119
fast block search, 56
Form-truing, 128
Function
Replacing, 281

## G

G code groups
Configuring the display, 48
G code program, 285
G group 59, 159
Generating a crashlog file, 346
Grinding parameters
Identifiers, 233

## H

HEAD_1, 143
HEAD_2, 144
High-Speed Settings, 159
Hirth gearing, 131
HMI trace, 352
HOLES2, 92

## I

Identifier
Coolant, 271
Cutting edge parameters, 230
Grinding parameters, 233
Magazine location adapter parameters, 235
Magazine location parameters, 234
Monitoring parameters, 232
Multitool location parameter, 236
Multitool parameters, 235
OEM cutting edge parameter, 269
OEM magazine location parameters, 270
OEM monitoring parameters, 270
OEM multitool parameters, 270
OEM tool parameters, 269
Tool parameters, 228
Tool-specific functions, 271
Inclined Y axis
under turning, 122
Inserting a run up screen, 19

## K

Keyboard
Setting, 19
Kinematics
Check list, 135
Examples, commissioning, 143
measured, 157
with Hirth gearing, 131

## L

Language
Defining the selection mode, 18
List parameters
Changing, 237
Creating new, 239

Log file
Action log, 349

## M

Machine data
Configuring plain texts, 25
Machine function, defining texts, 260
Machine operating area, 47
Magazine location adapter parameters
Identifiers, 235
Magazine location parameters
Identifiers, 234
OEM identifier, 270
Manual machine
Activating the function, 56
Parameter assignment, 56
Milling
Cylinder surface transformation, 95
Displaying softkeys, 71
Examples, setting-up a milling machine, 96
Settings under ShopMill, 93
Tool measurement, 170
Workpiece measurement, 168, 181
MIXED_45, 147
Monitoring parameters
Identifiers, 232
OEM identifier, 270
Multiple clamping of different workpieces (option), 99
Multiple edge, 92
Multitool
Loading and unloading, 287
Tool change, 289
Multitool location parameter
Identifiers, 236
Multitool parameters
Identifiers, 235
OEM identifier, 270

## N

NC/PLC variables
Changing, 303
Displaying, 301
Network
Configuring, 35
Network diagnostics, 38
Network interfaces
Configuring, 39

## O

Operating state, 305

## P

Password
Changing, 10
Delete, 10
Set, 10
Probe
Example of a test program, 162
Testing the function, 162
to measure a tool, 161
to measure the workpiece, 161
PROFINET
Connections, 321
Devices, 323
PROG_EVENT.SPF, 78, 321
Program list
Configuring a softkey, 31
Creating, 32
Program test
activate, 53
deactivate, 53
Programming G code programs, 60

## R

Replacing the logo, 47

## S

Screenshots
Copy, 20
Creating, 20
Search mode
Configuring, 55
Service drive
Display, 313
Parameter description, 314
Service overview, 305
Axes and drives, 307
Setting up a counterspindle
ShopTurn, 107
Setting-up a lathe with milling tools, 114
Setting-up measuring in JOG, 166

ShopMill
Settings for milling, 93
Tool-specific functions, 259
ShopTurn
Setting up a counterspindle, 107
Settings for turning, 102
Simulation
Deactivate, 64
General conditions, 61
Grinding technology, 63
Milling technology, 63
Turning technology, 63
Simultaneous recording, 65
Activating the machining time, 66
Option, 65
slmagcodeconfig.ini, 48
SLOT2, 92
Softkey
Access level, 15
Original setting, 15
Spindle analog, 197
Spindle control, 195
Spindle diagnostics, 198
Additional temperature sensor, 202
Clamping state, 204
Clamping time statistics, 208
Diagnostic statistics, 207
Logistics data, 208
Motor temperature sensor, 202
Speed limitation, 206
Spindle load, 203
Temperature, 200
Temperature histograms, 203
Station-related network diagnostics, 38
Status display
Inserting a logo, 47
Stock removal, corner, 111
Surface grinding, 127
Displaying softkeys, 71
Swiveling
activate, 129
Commission kinematic chain, 135
Configuring the input dialog, 129
Offset vectors, 137
Retraction, 142
Rotary axes, 139

## T

TABLE_45, 146
TABLE_5, 149
Tailstock
Option, 65
Tapping, 76
TEACH IN, 54
Technology
Cylindrical grinding, 127
Drilling, 75
Milling, 91
Surface grinding, 127
Swiveling, 128
Turning, 110
Thread-cutting, 110
Time
Selecting the format, 17
Setting, 17
TO units
Assigning, 251
Tool holder
Assigning, 251
Tool list
List views, 224
Tool management
Enable functions, 214
Graphical tool/magazine display, 215
Grinding technology, 218
Manual tool, 216
MD settings, 211
MD settings for diameter / cutting edge radius, 214
Milling technology, 218
PLC,TRANSLINE 2000, 218
Tool wear, 216
Turning technology, 218
With magazine management, 212
Without magazine management, 211
Tool Management
Assignments, 299
Buffer locations, 296
Editing the management, 293
Loading locations, 297
Magazine configuration, 298
Opening a function, 292
Tool unit, 295
Tool measurement
under milling, 170
under turning, 175
Tool parameters
Identifiers, 228
OEM identifier, 269
Tool types
Drill, 241
Grinding tools, 241

Milling tool, 241
Special tools, 242
Turning tools, 242
Toolholder, 284
Tool-specific functions
Identifier, 271
Topologies, 43
TRAANG, 124
under turning, 122
Trace
Changing the scale, 339
Creating a session, 326
Cursor A/B positioning, 341
Data acquisition, 334
Data storage on CU, 335
Define zoom area, 341
Display peak-to-peak value, 342
Display peak-to-valley value, 342
Displaying variable details, 333
Editing a session, 330
Editing graphic display, 337
Filter/search, 329
Insert variable, 329
Load session, 328
Overview, 324
Replace variable, 330
Save options, 336
Save session, 327
Selecting a variable, 329, 338
Sequence, 326
Session, 329
Setting grid lines, 336
Settings (drive), 334
Settings (PLC, NC, servo), 334
Start recording, 336
Starting a session, 328
Stop recording, 336
Variable attributes, 330
Zooming characteristic graphs, 340
TRACON, 124
TRACYL, 124
under turning, 115
Transformation
Settings for milling, 97
TRANSMIT
under turning, 119
with Y axis, 120
TRANSMIT command, 285
Turning
Cylinder surface transformation, 115
Displaying softkeys, 71
Examples, setting-up a lathe, 115
Face end machining, 119
Inclined Y axis, 122
Settings under ShopTurn, 102
Technology cycles, 110
Tool measurement, 175
Workpiece measurement, 188
Zero point of the X-axis displaced, 125

## U

User view
Creating, 22
Delete, 25
Editing, 23
Importing, 21

## V

Variable screen forms, 304

## W

Workpiece measurement
under milling, 168, 181
under turning, 188

## Z

Zero point displaced
under turning, 125

SIEMENS

Generating user dialogs 1

Generating commissioning dialogs 2

# SINUMERIK

# SINUMERIK 840D sl/828D Easy XML

Programming Manual

Valid for:

CNC Software Version 4.7 SP1

01/2015
6FC5397-3DP40-5BA2

## Legal information

### Warning notice system

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

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

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

| ⚠ CAUTION |
| --- |
| 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.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

Order number: 6FC5397-3DP40-5BA2
℗ 02/2015 Subject to change

Copyright © Siemens AG 2014.
All rights reserved

# Table of contents


**1 Generating user dialogs....5**

1.1 Scope of functions....5

1.2 Fundamentals of Configuration....7

1.3 Configuration files....9

1.4 Structure of configuration file....13

1.5 Language dependency....15

1.6 XML identifier....15
1.6.1 General structure....15
1.6.2 Instruction/identifier descriptions....16
1.6.3 Color coding....40
1.6.4 Special XML syntax....40
1.6.5 Operators....41
1.6.6 Generating softkey menus and dialog forms....42

1.7 Generating user menus....68
1.7.1 Creating processing cycle forms....68
1.7.2 Substitution characters....71

1.8 Addressing components....71
1.8.1 PLC addressing....71
1.8.2 Addressing NC variables....73
1.8.3 Generating NC/PLC addresses during the runtime....73
1.8.4 Addressing drive components....73
1.8.5 Addressing machine and setting data....76
1.8.6 Addressing user data....77

1.9 Predefined functions....78

**2 Generating commissioning dialogs....111**

2.1 Overview of functions....111

2.2 Configuration in the PLC user program....114

2.3 Display on the user interface....116

2.4 Creating language-dependent texts....117

2.5 User example for a power unit....118

2.6 Script language....119
2.6.1 CONTROL_RESET....122
2.6.2 FILE....122
2.6.3 OPTION_MD....123
2.6.4 PLC_INTERFACE....124
2.6.5 POWER_OFF....125
2.6.6 WAITING....125
2.6.7 XML identifiers for the dialog....125

2.6.8 SOFTKEY_OK, SOFTKEY_CANCEL....126
**Index....129**

# Generating user dialogs

# 1

## 1.1 Scope of functions

### Overview

The "Generate user dialogs" function offers an open structure and enables the user to develop customer-specific and application-specific HMI interfaces.

The control system offers an XML-based script language for generating user dialogs.

This script language makes it possible to display machine-specific menus and dialog forms in the <CUSTOM> operating area on the HMI.

All dialog forms can be designed on a language-neutral basis. In such cases, the system reads out the texts to be displayed from the accompanying language database.

### Use

The defined XML instructions offer the following properties:

1. Display dialogs containing the following elements:
   - Softkeys
   - Variables
   - Texts and Help texts
   - Graphics and Help displays
2. Call dialogs by:
   - Pressing the (start) softkeys
3. Restructure dialogs dynamically:
   - Edit and delete softkeys
   - Define and design variable fields
   - Insert, exchange, and delete display texts (language-dependent or language-neutral)
   - Insert, exchange, and delete graphics
4. Initiate operations in response to the following actions:
   - Displaying dialogs
   - Inputting values (variables)
   - Selecting a softkey
   - Exiting dialogs
5. Data exchange between dialogs

6. Variables
   - Read (NC, PLC and user variables)
   - Write (NC, PLC and user variables)
   - Combine with mathematical, comparison or logic operators
7. Execute functions:
   - Subprograms
   - File functions
   - PI services
8. Apply protection levels according to user classes

The valid elements (tags) for the script language are described in the "XML tags" (Page 15) section.

**Note**

The following section is not intended as a comprehensive description of XML (Extensible Markup Language). Please refer to the relevant specialist literature for additional information.

# 1.2 Fundamentals of Configuration

## Configuration files

The defining data for new user interfaces are stored in configuration files. These files are automatically interpreted and the result displayed on the screen. Configuration files are not stored in the software supplied and must first be set up and loaded by the user.

An XML editor or another form of text editor can be used to generate the configuration files.

**Note**

**Important:**

All file names must be written in lowercase.

## Menu tree principle

Several interlinked dialogs create a menu tree. A link exists if you can switch from one dialog to another. You can use the newly defined horizontal/vertical softkeys in this dialog to call the preceding or any other dialog.

Configured start softkeys can be used to create a further menu tree behind the start menu:

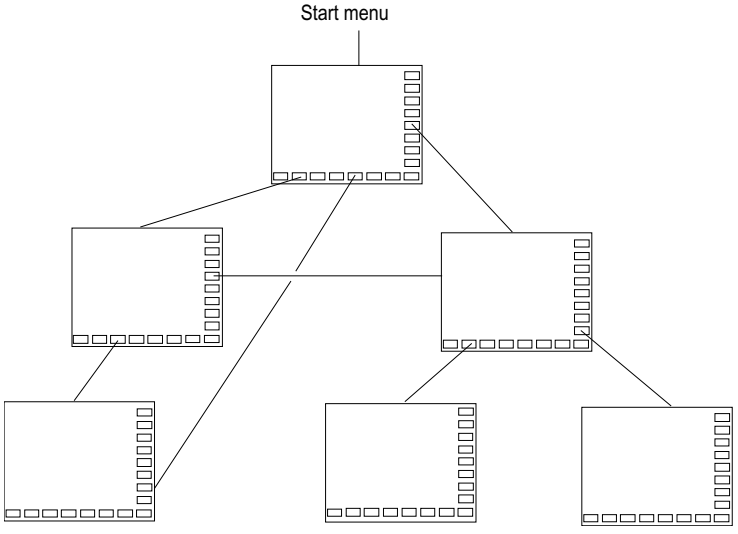



Figure 1-1 Menu tree for user dialogs

## Start menu

The start menu is defined by the name "main" in the "xmldial.xml" file. The start menu is used to initiate your own operating sequences.

You can use the "main" menu to link your own dialogs or additional softkey bars so that they can be loaded and used for executing additional actions.

The figure below shows the manufacturer's drive on the control system.

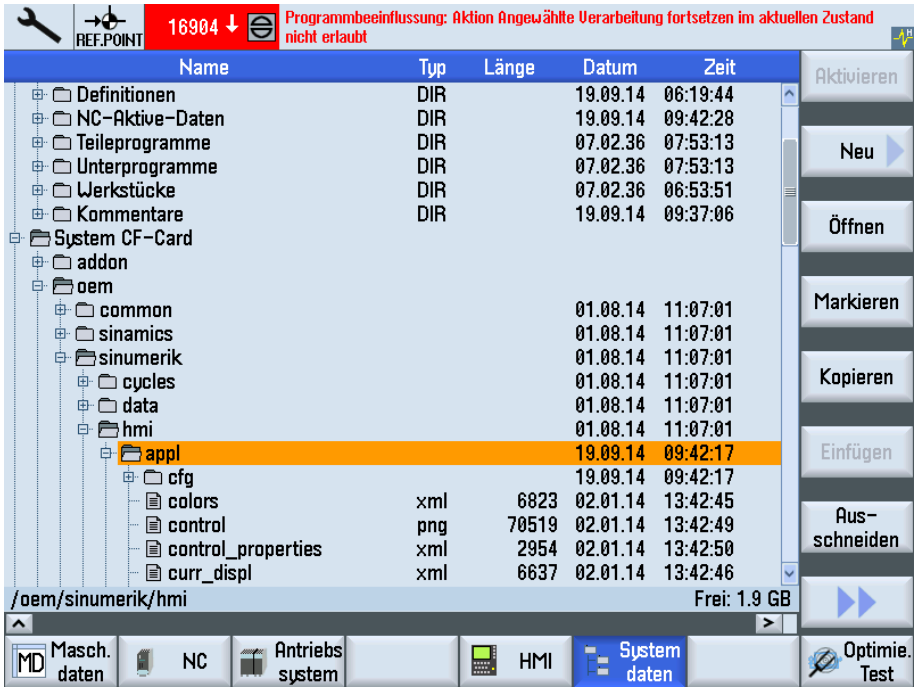



Figure 1-2 Manufacturer's drive

The following files in the control system's "Manufacturer's drive" directory are needed to configure the user dialogs:

Table 1-1 Files for configuration

| File type | Name of the file | Description | Storage location in the HMI operating area Commissioning |
|---|---|---|---|
| Script file | "xmldial.xml" | This script file uses XML tags to control the process image of the configured softkey menus and dialog forms in the <CUSTUMER> operating area on the HMI. | "Systemcard" > "oem/sinumerik/hmi" > in the "appl" subdirectory for the applications |
| Text file | "oem_xml_screens_xxx.ts" | This text file contains the texts for the menus and dialog forms for the individual languages. | "Startup files" > "Manufacturer's drive" > in the "lng" subdirectory for the desired languages |
| Bitmaps | | The control system supports BMP and PNG formats. | "Startup files" > "Manufacturer's drive" > in the "ico" subdirectory |
| XML files inserted in the "xmldial.xml" control file with the "INCLUDE" XML tag. | E.g. "machine_settings.xml" | These files also contain programmed instructions for displaying the dialog forms and parameters on the HMI. | "Startup files" > "Manufacturer's drive" > in the "appl" subdirectory for the applications |

# 1.3 Configuration files

## Introduction

The screen below shows the manufacturer's drive on the control system.

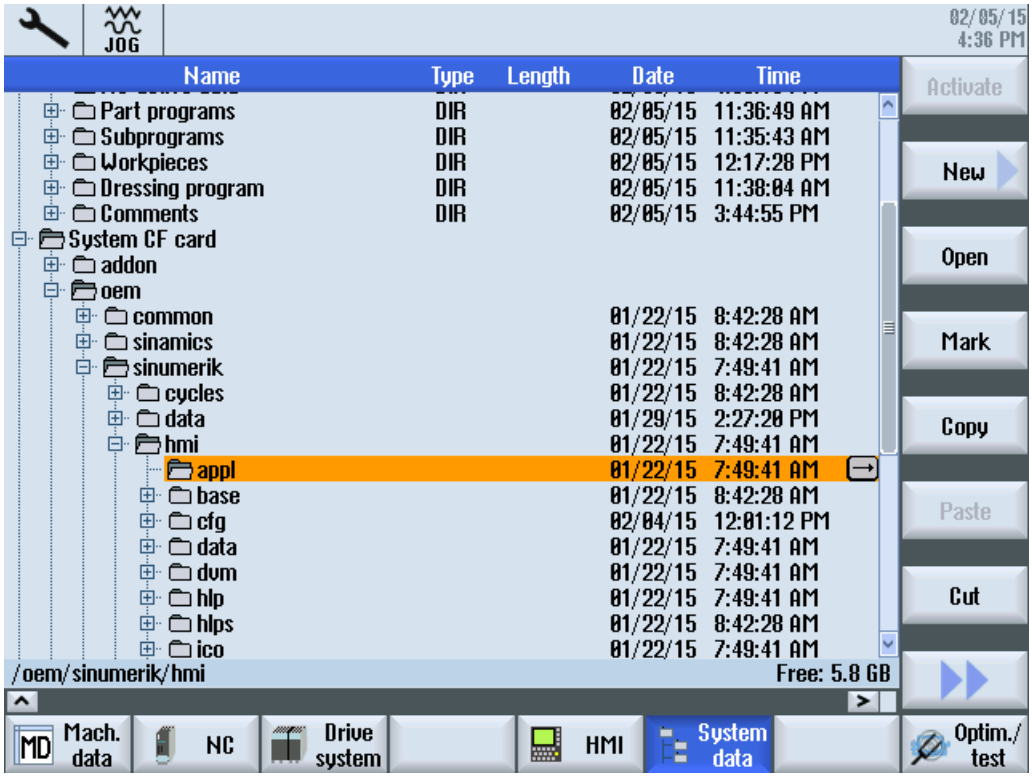



Figure 1-3 Manufacturer's drive

The following files in the control system's "Manufacturer's drive" directory are needed to configure the user dialogs:

Table 1-2 Files for configuration

| File type | Name of the file | Meaning | Storage location in the HMI's <SYSTEM> or <PROGRAM MANAGER> operating area |
|---|---|---|---|
| Script file | "xmldial.xml" | This script file uses XML tags to control the process image of the configured softkey menus and dialog forms in the <COSTUMER> operating area on the HMI. | "Startup files" > "Manufacturer's drive" > in the "appl" subdirectory for the applications |
| Text file | "oem_xml_screens_xxx.ts" | This text file contains the texts for the menus and dialog forms for the individual languages. | "Startup files" > "Manufacturer's drive" > in the "lng" subdirectory for the languages |
| Bitmaps | | The control system supports BMP and PNG formats. | "Manufacturer's drive" > in the "ico" subdirectory<br>The bitmaps are saved in the subdirectory for the screen resolution belonging to the control.<br>Note: If a path to the bitmap file is specified, the files can be stored in this directory directly. |
| XML files inserted in the "xmldial.xml" control file with the "INCLUDE" XML tag. | E.g. "machine_settings.xml" | These files also contain programmed instructions for displaying the dialog forms and parameters on the HMI. | "Manufacturer's drive" > in the "appl" subdirectory for the applications |

## Dependencies of files for configuring user dialogs

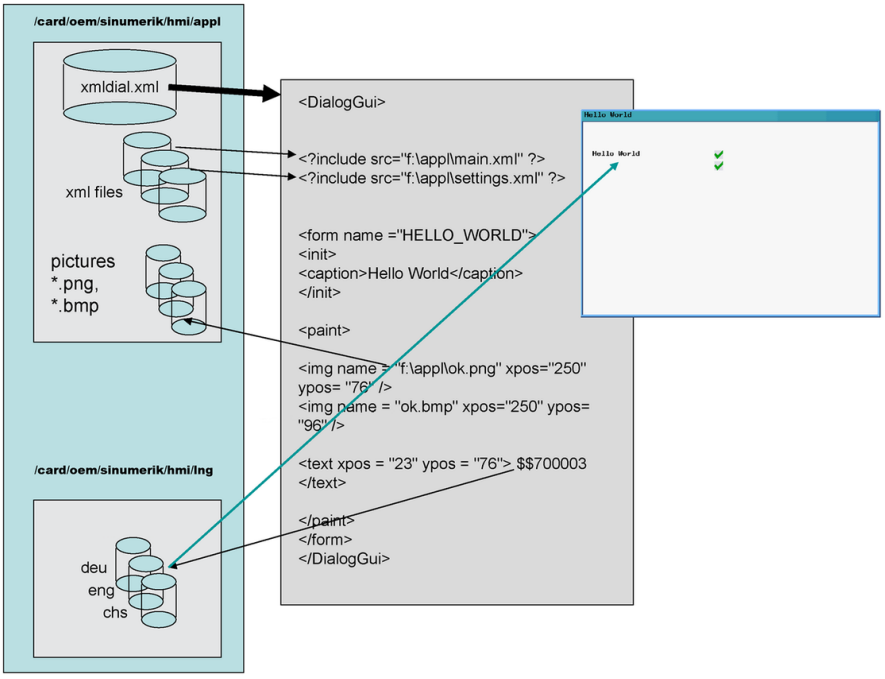



Figure 1-4 Dependencies

## Load configuration

As described in the "Storage location in the HMI" column in the previous "Files for configuration" table, the generated files must be copied to the relevant subdirectories in the "Manufacturer's drive" menu.

### Note

As soon as there is an "xmldial.xml" script file in the subdirectory for applications on the manufacturer's drive, the user can start this user dialog in the <CUSTOM> operating area.

After the initial copying process, the control system needs to be reset via "Normal power-up".

## Example of a user dialog on the HMI

The configured softkey menus are displayed when the <CUSTOM> operating area is called. This enables the user to operate the dialog forms which have been configured.

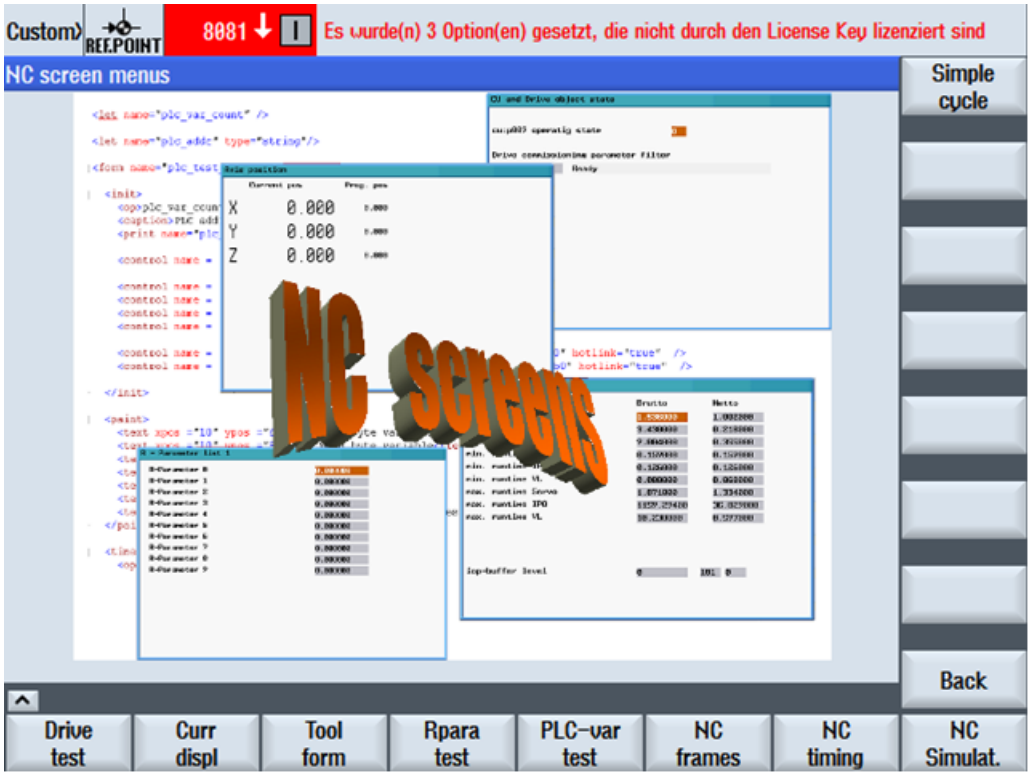



Figure 1-5 Example of a user dialog in the <CUSTOM> operating area

## See also

Predefined functions (Page 78)

# 1.4 Structure of configuration file

## Overview

A configuration file consists of the following elements:

- Description of the "main" start menu with start softkeys
- Definition of dialogs
- Definition of variables
- Description of the blocks
- Definition of softkey bars

The following screens show an XML script for the "xmldial.xml" file and the corresponding screenshots.

The script contains the dialogs for displaying actual values and residual distances, as well as an R parameter list.

```
<DialogGui>

<!--
main menu
It is called by the system software. It starts the application.
The menu tag manages the soft key reactions. One input form can be assigned to a menu tag.
-->

<menu name = "MAIN">
<OPEN_FORM name = "CURRENT_DISPLAY" />

<softkey POSITION="1">
<caption>R-%nPrameter</caption>
<navigation>MENU_R_PARAMETER</navigation> <!-- opens the menu R parameter -->
</softkey>


</menu>


<form name = "CURRENT_DISPLAY">

 <init>

 <caption>Istwerte</caption>

 <control name = "label1" xpos = "36" ypos = "56" width = "32" fieldtype="readonly" refvar="nck/Channel/GeometricAxis/name[0]" />
 <control name = "label2" xpos = "36" ypos = "76" width = "32" fieldtype="readonly" refvar="nck/Channel/GeometricAxis/name[1]" />
 <control name = "label3" xpos = "36" ypos = "96" width = "32" fieldtype="readonly" refvar="nck/Channel/GeometricAxis/name[2]" />
 <control name = "label4" xpos = "36" ypos = "116" width = "32" fieldtype="readonly" refvar="nck/Channel/GeometricAxis/name[3]" />
 <control name = "label5" xpos = "36" ypos = "136" width = "32" fieldtype="readonly" refvar="nck/Channel/GeometricAxis/name[4]" />


 <control name = "edit1" xpos = "80" ypos = "56" refvar="nck/Channel/GeometricAxis/actProgPos[0]" hotlink="true" fieldtype="readonly" format="%9.3f" time="super fast"/>
 <control name = "edit2" xpos = "80" ypos = "76" refvar="nck/Channel/GeometricAxis/actProgPos[1]" hotlink="true" fieldtype="readonly" format="%9.3f" time="super fast"/>
 <control name = "edit3" xpos = "80" ypos = "96" refvar="nck/Channel/GeometricAxis/actProgPos[2]" hotlink="true" fieldtype="readonly" format="%9.3f" time="super fast"/>
 <control name = "edit4" xpos = "80" ypos = "116" refvar="nck/Channel/GeometricAxis/actProgPos[3]" hotlink="true" fieldtype="readonly" format="%9.3f" time="super fast"/>
 <control name = "edit5" xpos = "80" ypos = "136" refvar="nck/Channel/GeometricAxis/actProgPos[4]" hotlink="true" fieldtype="readonly" format="%9.3f" time="super fast"/>

 <control name = "edit11" xpos = "210" ypos = "56" refvar="nck/Channel/GeometricAxis/progDistToGo[0]" hotlink="true" fieldtype="readonly" format="%9.3f" time="super fast"/>
 <control name = "edit12" xpos = "210" ypos = "76" refvar="nck/Channel/GeometricAxis/progDistToGo[1]" hotlink="true" fieldtype="readonly" format="%9.3f" time="super fast"/>
 <control name = "edit13" xpos = "210" ypos = "96" refvar="nck/Channel/GeometricAxis/progDistToGo[2]" hotlink="true" fieldtype="readonly" format="%9.3f" time="super fast"/>
 <control name = "edit14" xpos = "210" ypos = "116" refvar="nck/Channel/GeometricAxis/progDistToGo[3]" hotlink="true" fieldtype="readonly" format="%9.3f" time="super fast"/>
 <control name = "edit15" xpos = "210" ypos = "136" refvar="nck/Channel/GeometricAxis/progDistToGo[4]" hotlink="true" fieldtype="readonly" format="%9.3f" time="super fast"/>

 </init>

 <paint>
 <text xpos= "36" ypos="30">Name</text>

 <text xpos= "80" ypos="30">Current pos</text>
 <text xpos= "210" ypos="30">Prog. pos</text>

 </paint>

</form>

<!-- ***************************************************************
Menu R- Parameter
-->

<menu name ="MENU_R_PARAMETER">
<OPEN_FORM name = "R-Parameter" />

<softkey POSITION="16">
<caption>Back</caption>
<navigation>MAIN</navigation>
</softkey>

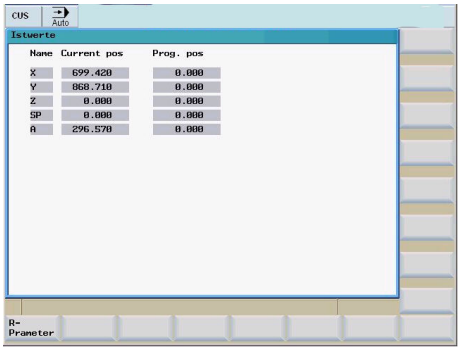
</menu>

<form name = "R-Parameter">

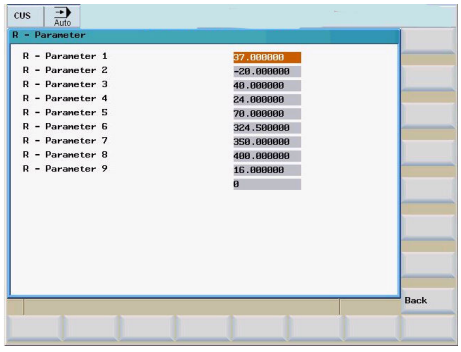
 <init>
 <DATA_ACCESS type="true" />
 <caption>R - Parameter</caption>
 <control name = "edit1" xpos = "322" ypos = "34" refvar="nck/Channel/Parameter/R[1]" />
 <control name = "edit2" xpos = "322" ypos = "54" refvar="nck/Channel/Parameter/R[2]" />
 <control name = "edit3" xpos = "322" ypos = "74" refvar="nck/Channel/Parameter/R[3]" />
 <control name = "edit4" xpos = "322" ypos = "94" refvar="nck/Channel/Parameter/R[4]" />
 <control name = "edit5" xpos = "322" ypos = "114" refvar="nck/Channel/Parameter/R[5]" />
 <control name = "edit6" xpos = "322" ypos = "134" refvar="nck/Channel/Parameter/R[6]" />
 <control name = "edit7" xpos = "322" ypos = "154" refvar="nck/Channel/Parameter/R[7]" />
 <control name = "edit8" xpos = "322" ypos = "174" refvar="nck/Channel/Parameter/R[8]" />
 <control name = "edit9" xpos = "322" ypos = "194" refvar="nck/Channel/Parameter/R[9]" />
 <control name = "edit10" xpos = "322" ypos = "214" refvar="nck/Channel/Parameter/R[10]" />
 <control name = "edit10" xpos = "322" ypos = "214" refvar="plc/mb170" />
 </init>

 <paint>
 <text xpos = "23" ypos = "34">R - Parameter 1</text>
 <text xpos = "23" ypos = "54">R - Parameter 2</text>
 <text xpos = "23" ypos = "74">R - Parameter 3</text>
 <text xpos = "23" ypos = "94">R - Parameter 4</text>
 <text xpos = "23" ypos = "114">R - Parameter 5</text>
 <text xpos = "23" ypos = "134">R - Parameter 6</text>
 <text xpos = "23" ypos = "154">R - Parameter 7</text>
 <text xpos = "23" ypos = "174">R - Parameter 8</text>
 <text xpos = "23" ypos = "194">R - Parameter 9</text>
 </paint>
</form>

</DialogGui>
```




Actual values




R parameters

Figure 1-6 XML script and dialog forms with parameters

## 1.5 Language dependency

Language-dependent texts are used for:

- Softkey labels
- Headers
- Help texts
- Any other texts

The language-dependent texts are stored in text files.

**Note**

You will need to perform the following steps when using these text files:

- Make them available in the required languages.
- Transfer them into the relevant language directories of the control system.

## 1.6 XML identifier

### 1.6.1 General structure

#### Structure and instructions of the script file for dialog configuration

All dialog configurations should be stored in the **DialogGui** tag.

```
<DialogGui>
…
</DialogGui>
```

**Example:**

```
<?xml version="1.0" encoding="utf-8"?>
<DialogGui>
…
<FORM name ="Hello_World">
<INIT>
<CAPTION>Hello World</CAPTION>
</INIT>
...
</FORM>

</DialogGui>
```

### Instructions

The language offers the following instructions for executing conditional instructions and loop controls:

- For loop
- While loop
- Do with loop
- Conditional processing
- Switch and case instructions
- Operator controls in a dialog form
- Softkey descriptions
- Define variables

For a detailed description of instructions, see Instruction/identifier descriptions (Page 16).

## 1.6.2 Instruction/identifier descriptions

The following **XML tags** are defined for generating dialogs and menus, and for executing program sequences:

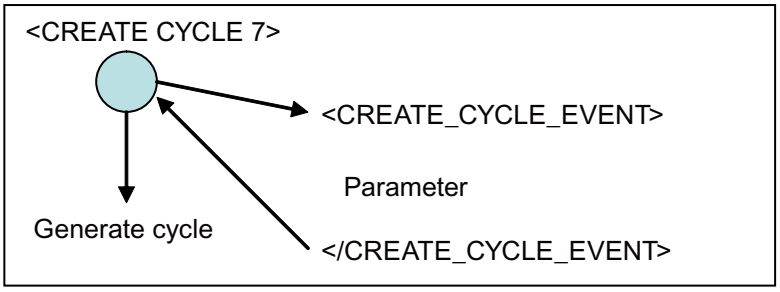
**Note**

Attribute values that are in quotation marks "<...>" should be replaced by the currently used expressions.

**Example:**
<DATA_LIST action="read/write/append" id="<list name>">
is programmed as follows:

<DATA_LIST action="read/write/append" id="my datalist">

| Tag identifier | Meaning |
|---|---|
| **BREAK** | Conditional cancellation of a loop. |
| **CONTROL_RESET** | The tag enables one or more control components to be restarted.<br><br>**Syntax:**<br>`<CONTROL_RESET resetnc="TRUE" />`<br><br>**Attributes:**<br>• **RESETNC = "TRUE"**<br>The NC component is restarted.<br>• **RESETDRIVE = "TRUE"**<br>The drive components are restarted. |

| Tag identifier | Meaning |
|---|---|
| CREATE_CYCLE_EVENT | If the parser starts to process the tag CREATE_CYCLE, initially, the message <CREATE_CYCLE_EVENT sent> is sent to the active form. This message can be used for preparing the cycle parameters, before the parser generates the NC operation from the parameter list and the generation rule.<br><br><br>**Syntax:**<br>`<CREATE_CYCLE_EVENT>`<br>…<br>…<br>`</CREATE_CYCLE_EVENT>`<br><br>**Example:**<br>`<SOFTKEY_OK>`<br>…<br>`<CREATE_CYCLE />`<br>…<br>…<br>`<SOFTKEY_OK>`<br>…<br>…<br><br>`<FORM>`<br>`<NC_INSTRUCTION>MY_CYCLE($P1, $P2)</ NC_INSTRUCTION >`<br>…<br>…<br>`<CREATE_CYCLE_EVENT>`<br>`<type_cast name="P1" type="int"/>`<br><br>`<OP> P1 = P1 * 150 </OP>`<br><br>`</CREATE_CYCLE_EVENT>`<br>…<br>…<br><br>`</FORM>` |

| Tag identifier | Meaning |
|---|---|
| **DATA** | The tag enables the NC, PLC, GUD and drive data to be directly written to.<br>The "Component addressing" (Page 71) section contains details on address formation.<br><br>**Attribute:**<br>• **name**<br>Variable address<br><br>**Tag value:**<br>All alphanumeric terms are approved as tag values. If a value is to be written from a local variable directly, the **$** replacement operator preceding the name of the local variable should be used.<br><br>**Syntax:**<br>`<DATA name="<variable name>"> value </DATA>`<br><br>**Example:**<br>`<DATA name = "plc/mb170"> 1 </DATA>`<br>…<br>`<LET name = "tempVar"> 7 </LET>`<br>`<!– the contents of the local variables "tempVar" are written to bit memory byte 170 →`<br>`<DATA name = "plc/mb170">$tempVar</DATA>` |
| **DATA_LIST** | The tag enables the listed drive and machine data to be saved or restored.<br>Addresses are listed in lines. The "Component addressing" (Page 71) section contains details on address formation.<br>Up to 20 temporary data lists can be created.<br><br>**Attributes:**<br>• **action**<br>*read* – the values of the listed variables are stored in a temporary memory<br>*append* – the values of the listed variables are added to an existing list<br>*write* – the backed up values are copied to the relevant machine data<br>• **id**<br>The identifier identifies the temporary memory<br>**Syntax:**<br>`<DATA_LIST action="<read/write/append>" id="<list name>">`<br>`NC/PLC address compilation`<br>`</DATA_LIST>`<br><br>**Example:**<br>`<DATA_LIST action ="read" id="<name>">`<br>`nck/channel/parameter/r[2]`<br>`nck/channel/parameter/r[3]`<br>`nck/channel/parameter/r[4]`<br>`$MN_USER_DATA_INT[0]`<br>…<br>`</ DATA_LIST>`<br>`<DATA_LIST action ="write" id="<name>" />` |

| Tag identifier | Meaning |
|---|---|
| **DYNAMIC_INCLUDE** | The tag includes an XML script file.<br>Contrary to the INCLUDE tag, read-in is first only realized when executing the corresponding operation.<br>For large projects, the use of the tag reduces the load time of the customer area and/or the cycle support. Further, the average level of resources is reduced, as not all of the dialogs are always called during a session.<br><br>**Syntax:**<br><DYNAMIC_INCLUDE src="path name"/><br><br>**Example:**<br>`<SOFTKEY POSITION="3">`<br>`<CAPTION>MY_MENU</CAPTION>`<br>`<DYNAMIC_INCLUDE src/>`<br>`<NAVIGATION>MY_MENU</NAVIGATION>`<br></SOFTKEY> |
| **ELSE** | Instruction for situations where the condition has not been met (**IF, THEN, ELSE**) |
| **FORM** | The tag contains the description of a user dialog. The relevant tags are described in the section on generating menus and dialog forms.<br><br>**Syntax:**<br>`<FORM name="<dialog name>" color="#ff0000">`<br><br>**Attributes:**<br>• **color**<br>Background color of the dialog form (color coding, see Chapter Color coding (Page 40))<br>– Default white<br>• **name**<br>Identifier of the form<br>• **type**<br>Permissible value is *cycle* , which identifies a user cycle screen form<br>• **xpos**<br>X-position of the top left corner of the dialog box (optional)<br>• **ypos**<br>Y position of the top left corner (optional)<br>• **width**<br>Extension in the X direction (in pixels) (optional)<br>• **height**<br>Extension in the Y direction (in pixels) (optional) |
| **HMI_RESET** | The tag initiates an HMI restart.<br>The interpretation is cancelled after this operation. |

| Tag identifier | Meaning |
|---|---|
| **IF** | Conditional instruction (**IF, THEN, ELSE**)<br>The THEN and ELSE tags are enclosed in the IF tag.<br>The condition that is executed in the CONDITION tag follows the IF tag. The further processing of the instructions depends upon the result of the operation. If the function result is true, then the THEN branch is executed and the ELSE branch is skipped. If the result of the function is false, the parser executes the ELSE branch.<br><br>**Syntax:**<br>`<IF>`<br>`<CONDITION> Condition != 7 </CONDITION>`<br>`<THEN>`<br>`Instruction for the case: Condition fulfilled`<br>`</THEN>`<br>`<ELSE>`<br>`Instruction for the case: Condition not fulfilled`<br>`</ELSE>`<br>`</IF>`<br><br>**Example:**<br>`<IF>`<br>`<CONDITION> "plc/mb170" != 7 </CONDITION>`<br>`<THEN>`<br>`<OP> "plc/mb170" = 7 </OP>`<br>…<br>`</THEN>`<br>`<ELSE>`<br>`<OP> "plc/mb170" = 2 </OP>`<br>…<br>`</ELSE>`<br>`</IF>` |
| **INCLUDE** | The instruction includes an XML description.<br>(see also DYNAMIC_INCLUDE in this table)<br><br>**Attribute:**<br>• **src**<br>Contains the path name.<br><br>**Syntax:**<br>`<?INCLUDE src="<Path name>" ?>` |

| Tag identifier | Meaning |
|---|---|
| **LET** | The instruction creates a local variable under the specified name.<br>**Fields:**<br>Using the attribute **dim** (dimension) single or two-dimensional fields can be created. The field index addresses the individual field elements.<br>For a two-dimensional field, initially the line index is specified and then the column index.<br>• Single-dimensional field:<br>Indices 0 to 4<br>0<br>1<br>2<br>3<br>4<br>• Two-dimensional field:<br>Index line 0 to 3 and index column 0 to 5<br>0,0 0,1 0,2 0,3 0,4 0,5<br>1,0 1,1 1,2 1,3 1,4 1,5<br>2,0 2,1 2,2 2,3 2,4 2,5<br>3,0 3,1 3,2 3,3 3,4 3,5<br>**Attributes:**<br>• **name**<br>Variable name<br>• **type**<br>The variable type can be an integer (INT), double (DOUBLE), float (FLOAT) or string (STRING). If there is no type instruction specified, the system creates an integer variable.<br>`<LET name = "VAR1" type = "INT" />`<br>• **permanent**<br>If the attribute is set to **true**, then the variable value is saved permanently. This attribute is only effective for a global variable.<br>• **dim**<br>The following number of field elements must be specified. For a two-dimensional field, the second dimension is specified after the first dimension separated by a comma.<br>A field element is accessed via the field index, which is specified in square brackets after the variable name.<br>name[index] or name[row,column]<br>– Single-dimensional field: dim="<Number of elements>"<br>– Two-dimensional field: dim="<Number of lines>,<number of columns>"<br>Non-initialized field elements are pre-assigned with "0". |

| Tag identifier | Meaning |
|---|---|
| **LET** Continued | **Example:**<br>Single-dimensional field:<br>`<let name="array" dim="10"></let>`<br>Two-dimensional field:<br>`<let name="list_string" dim="10,3" type="string"></let>`<br><br>**Pre-assignment:**<br>A variable can be initialized with a value.<br>`<LET name = "VAR1" type = "INT"> 10 </LET>`<br><br>If values comprising NC or PLC variables are saved in a local variable, the assignment operation automatically adapts the format to that of the variables which have been loaded.<br>• Pre-assignment for a string variable:<br>Texts containing more than one line can be assigned to a string variable if the formatted text is transferred as a value. If a line is to end with a line feed <LF> (line feed) , the characters "\\n" should be added at the end of the line. |

| Tag identifier | Meaning |
|---|---|
| | <LET name = "text" type = "string"> F4000 G94\\n<br>G1 X20\\n<br>Z50\\n<br>M2\\n<br></LET>><br><br>**Fields (Arrays):**<br><let name="list" dim="10,3"><br>{1,2,3},<br>{1,20}<br></let><br><br><let name="list_string" dim="10,3" type="string"><br>{"text 10","text 11"},<br>{"text 20","text 21"}<br></let><br><br>**Assignment:**<br>Values made up of the machine data or subroutines can be assigned to a variable using the assignment operation "=".<br>A variable remains valid until the end of the higher-level XML block.<br>Variables which are to be available globally should be created directly after the **DialogGUi** tag.<br>The following must be observed for a dialog box:<br>• The message processing opens the corresponding tag.<br>• The tag is closed after the message has been executed.<br>• All variables within the tag are deleted when closing.<br><br>**Variable type struct**<br>This variable type contains a composition of variables that can be addressed using the structure name. A structure can contain all variable types and structures.<br>Within the structure, a variable is declared with the "element" tag. The attributes of the tags and the initialization correspond to the attributes and initialization of the let instruction.<br><br><let name="name" type="struct"><br><element name="<Name>" type="<Variable type>" /><br>…<br>…<br>…<br><br></let><br><br>Access to a variable of the structure is via the structure name and variable name. Both names are separated by a point operator. |

| Tag identifier | Meaning |
|---|---|
| | <op><br>Structure_name.variable_name = value;<br></op><br><br>**Example:**<br><let name="info" type="struct"><br><element name="id" type="int" /><br><element name="name" type="string" /><br><element name="phone" type="string" /><br></let><br><br><op><br>info.id = 1;<br>info.name = _T"my name";<br>info.phone<br>= _T"0034 45634";<br></op> |
| **MSG** | The operator component shows the message which is indicated in the tag.<br>If an alarm number is used, the dialog box displays the text which is saved for the number.<br><br>**Example:**<br><MSG text ="my message" /> |

| Tag identifier | Meaning |
|---|---|
| **MSGBOX** | The instruction opens a message box whose return value can be used for branching.<br><br>**Syntax:**<br>`<MSGBOX text="<Message>" caption="<caption>" retvalue="<variable name>" type="<button type>" />`<br><br>**Attributes:**<br>• **text**<br>Text<br>• **caption**<br>Header<br>• **retvalue**<br>Name of the variables to which the return value is copied:<br>1 – OK<br>0 – CANCEL<br>• **type**<br>Acknowledgment options:<br>"BTN_OK"<br>"BTN_CANCEL"<br>"BTN_OKCANCEL"<br>If an alarm number is used for the **"text"** or **"caption"** attribute, the message box displays the text which is saved for the number.<br><br>**Example:**<br>`<MSGBOX text="Test message" caption="Information" retvalue="result" type="BTN_OK" />` |

| Tag identifier | Meaning |
|---|---|
| **OP** | The tag executes the specified operations.<br>The operations listed in Chapter "Operators (Page 41) can be executed.<br>For the purpose of accessing the NC, PLC, and drive data, the complete variable name should be placed in **quotation marks**. The "Component addressing" (Page 71) section contains details on address formation.<br>`PLC: "PLC/MB170"`<br>`NC: "NC/Channel/…"`<br><br>**Example:**<br>`<LET name = "tmpVar" type="INT"> </LET>`<br>`<OP> tmpVar = "plc/mb170" </OP>`<br>`<OP> tmpVar = tmpVar *2 </OP>`<br>`<OP> "plc/mb170" = tmpVar </OP>`<br><br>More than one equation can be used within an operation tag. A semicolon marks the end of the instruction.<br><br>**Example:**<br>`<op>`<br>`x= x+1;`<br>`y = y+1;`<br>`</op>`<br><br>**Character string processing:**<br>The operation instruction is able to process character strings and assign the results to the string variable specified in the equation.<br>The identifier _T should be placed at the start as a means of identifying text terms. Formatting of variable values is also possible. The identifier _F should be placed at the start of the formatting regulation, followed by the format instruction. The address is then specified for the variable.<br><br>**Example:**<br>`<LET name="buffer" type="string"></LET>`<br>…<br>…<br>…<br>`<OP> buffer = _T"unformatted value R0= " + "nck/Channel/Parameter/R[0]" + _T" and " + _T"$$85051" + _T" formatted value R1 " + _F %9.3f"nck/Channel/Parameter/R[1]" </OP>` |

| Tag identifier | Meaning |
|---|---|
| **XML_PARSER** | The “XML_PARSER“ tag parses XML files.<br>The parser interprets an XML file and calls defined call-back functions. Each call-back function belongs to a predefined event. The programmer can process the XML data within this function.<br><br>**Predefined events:**<br>• start document<br>The parser opens the document and starts with the parsing.<br>• end document<br>The parser closes the document.<br>• start element<br>The parser has found an element and creates a list with all attributes and attribute values. These lists are forwarded to the call-back function.<br>• end element<br>The end of the element was found.<br>• characters<br>The parser forwards all characters of an element.<br>• error<br>The parser has detected a syntax error.<br>When an event occurs, the parser calls the call-back function and checks the function return value. If the function returns "true", the parser continues the process.<br><br>**Interfaces**<br>The value of the **name** attribute contains the path to the XML file.<br>To assign events to the call-back functions, the following properties must be specified:<br>**Default**<br>startElementHandler<br>endElementHandler<br>charactersHandler<br>**Optional**<br>errorHandler<br>documentHandler<br>The value of an attribute defines the name of the call-back function.<br><br>**Example:**<br>`<XML_PARSER name="f:\appl\xml_test.xml">`<br>`<!-- standard handler -->`<br>`<property startElementHandler="startElementHandler" />`<br>`<property endElementHandler="endElementHandler" />`<br>`<property charactersHandler="charactersHandler" />`<br>`<!-- optional handler -->`<br>`<property errorHandler="errorHandler" />`<br>`<property documentHandler="documentHandler" />`<br>`</XML_PARSER>` |

| Tag identifier | Meaning |
|---|---|
| **XML_PARSER**Continued | The parser also supplies variables so that the call-back functions can access the event data.<br><br>**startElementHandler:**<br>**Function parameters**<br>tag_name - tag name<br>num - number of found attributes<br>**System variables**<br>$xmlAttribute String array that contains the 0-num attribute name range.<br>$xmlValue String array that contains the 0-num attribute value range.<br><br>**Example:**<br>`<function_body name="startElementHandler" return="true" parameter="tag_name, num">`<br>`<print text="attribute name %s"> $xmlAttribute[0]</print>`<br>`<print text="attribute value %s"> $xmlValue [0]</print>`<br>`</function_body>`<br><br>**endElementHandler:**<br>**Function parameters**<br>tag_name - tag name<br><br>**Example:**<br>`<function_body name="endElementHandler" parameter="tag_name">`<br>`<print text="name %s"> tag_name </print>`<br>`</function_body>` |

| Tag identifier | Meaning |
|---|---|
| **XML_PARSER**Continued | **charactersHandler:**<br>**System variables**<br>$xmlCharacters String with data<br>$xmlCharactersStart Always 0<br>$xmlCharactersLength Number of bytes<br><br>**Example:**<br>`<function_body name="charactersHandler" return="true" >`<br>`<print text="chars %s"> $xmlCharacters </print>`<br>`</function_body>`<br><br>**documentHandler:**<br>**Function parameters**<br>state 1 start document, 2 end document<br><br>**errorHandler:**<br>**System variables**<br>$xmlErrorString Contains the invalid line (system variable)<br><br>**Example:**<br>`<function_body name="errorHandler" >`<br>`<print text="error %s">$xmlErrorString</print>`<br>`</function_body>`<br><br>**Call-back result:**<br>$return If 1 (true), the parser continues parsing the file |

| Tag identifier | Meaning |
|---|---|
| CONTROL | **Change control after the creation**<br>A control tag changes the control properties of an existing control after the creation. The tag must be specified with the name of the control to be changed and the new properties. It can be executed only within a form tag.<br>The following properties can be changed:<br>• xpos<br>• ypos<br>• width<br>• height<br>• color_bk<br>• color_fg<br>• access level<br>• fieldtype<br>• itemdata<br>• min.<br>• max<br>• default<br>The reference variable cannot be modified.<br>If a property should be changed by triggering caused by a softkey event, the send message tag must transfer this request to the form context. The tag message acquires the message.<br><br>**Example**<br>`<menu name = "main">`<br>`<open_form name = "attrib_form" />`<br>`<softkey POSITION="3">`<br>`<caption>Set%nro</caption>`<br>`<send_message>1, 0</send_message>`<br>`</softkey>`<br>`<softkey POSITION="4">`<br>`<caption>Set%nrw</caption>`<br>`<send_message>2, 0</send_message>`<br>`</softkey>`<br>`</menu>`<br>`<form name="attrib_form">`<br>`<init>`<br>`<control name="c_p0" xpos="60" ypos="70" width="272" />`<br>`</init>`<br>`<message>`<br>`<switch>`<br>`<condition>$message_par1</condition>`<br>`<case value="1">`<br>`<control name="c_p0" fieldtype="readonly" />`<br>`</case>`<br>`<case value="2">`<br>`<control name="c_p0" fieldtype="edity" />`<br>`</case>`<br>`</switch>`<br>`</message>`<br>`</form>` |

| Tag identifier | Meaning |
|---|---|
| OPERATION | **Operation**<br>Instructions can be moved within an equation.<br>**Move left "<<" operator**<br>The << function moves bits to the left. You can specify the value and the number of move increments directly or with a variable. If the limit of the data format is reached, the bits will be moved beyond the limit without an error message being issued.<br>**Execution rule**<br>Execution from left to right; '+' and '-' have priority over '<<' and '>>'<br>**Example:**<br>`<op> idx = idx << 2 </op>`<br>`<op> idx = 3 + idx << 2 </op>`<br>**Move right ">>" operator**<br>The >> function moves bits to the right. You can specify the value and the number of move increments directly or with a variable. If the limit of the data format is reached, the bits will be moved beyond the limit without an error message being issued.<br>**Execution rule**<br>Execution from left to right; '+' and '-' have priority over '<<' and '>>'<br>**Example:**<br>`<op> idx = idx >> 2 </op>`<br>`<op> idx = 3+idx >> 2</op>` |
| PASSWORD | The tag opens a dialog for entering the password.<br>Once the entry has been confirmed, the character string is available in the specified reference variable.<br>**Syntax:**<br>`<PASSWORD refVar ="<variable name>" />`<br>**Attribute:**<br>• **refVar**<br>Name of the reference variable<br>**Example:**<br>`<PASSWORD refvar="plc/mw107" />` |
| POWER_OFF | A message prompts the operator to switch the machine off. The message text is permanently saved in the system. |

| Tag identifier | Meaning |
|---|---|
| **PRINT** | The tag outputs a text in the dialog line or copies the text to the variable specified.<br>If the text contains formatting identifiers, the variable values are inserted at the appropriate places.<br><br>**Syntax:**<br>`<PRINT name="Variable name " text="text %Formatting "> Variable, … </PRINT>`<br>`<PRINT text="text %Formatting"> Variable, … </PRINT>`<br>**Attributes:**<br>• **name**<br>Name of the variable where the text is to be stored (optional)<br>• **text**<br>Text<br><br>**Formatting:**<br>The character "%" causes the variable specified as the value to be formatted.<br>%[Flags] [Width] [.decimal places] type<br>• Flags:<br>Optional character for defining output formatting:<br>– Right-justified or left-justified ("-") for left-justified<br>– Add leading zeros ("0")<br>– Fill with blanks<br>• Width:<br>The argument defines the minimum output width for a non-negative number. If the value to be output has fewer places than the argument defined, the missing spaces are filled with blanks.<br>• Decimal places:<br>With floating-point numbers, the optional parameter defines the number of decimal places.<br>• Type:<br>The type character defines which data formats are transferred for the print instruction. These characters need to be specified.<br>– d: Integer value<br>– f: Floating-point number<br>– s: String |

| Tag identifier | Meaning |
|---|---|
| **PRINT** Continued | **Values:**<br>Number of variables whose values are to be inserted into the text.<br>The variable types must match the corresponding type identifier for the formatting instruction and must be separated from one another with a comma.<br><br>**Example:**<br>Output of a text in the information line<br>`<PRINT text="Infotext" />`<br>Output of a text with variable formatting<br>`<LET name="trun_dir"></LET>`<br>`<PRINT text="M%d">trun_dir</PRINT>`<br>Output of a text in a string variable with variable formatting<br>`<LET name="trun_dir"></LET>`<br>`<LET name="str" type="string" ></LET>`<br>`<print name="str" text="M%d ">trun_dir</print>` |
| **PROGRESS_BAR** | The tag opens or closes a progress bar. The bar is displayed below the application window.<br><br>**Syntax:**<br>`<PROGRESS_BAR type="<true/false>"> value </ PROGRESS_BAR>`<br><br>**Attributes:**<br>• **type =** "TRUE" - opens the progress bar<br>• **type** = "FALSE" - closes the progress bar<br>• **min**<br>(optional) – minimum value<br>• **max**<br>(optional) – maximum value<br><br>**Value:**<br>• **Value**<br>Percentage position of the bar<br>**Example:**<br>`<PROGRESS_BAR type="true" min="0" max="101" >20< /`<br>`PROGRESS_BAR>......<PROGRESS_BAR >50< /`<br>`PROGRESS_BAR>......<PROGRESS_BAR type="false" >100< /PROGRESS_BAR>` |

| Tag identifier | Meaning |
|---|---|
| **SEND_MESSAGE** | The tag sends a message with two parameters to the active form, which is processed in the tag message.<br><br>**Syntax:**<br>`<SEND_MESSAGE>p1, p2</SEND_MESSAGE>`<br><br>**Example:**<br>`<SOFTKEY POSITION="3">`<br>`  <caption>Set%nParameter</caption>`<br>`  <send_message>1, 0</send_message>`<br>`</SOFTKEY>`<br><br>`<FORM>`<br>…<br>…<br>`<MESSAGE>`<br>`<SWITCH>`<br>`<CONDITION>$message_par1</CONDITION>`<br>`<CASE value="1">`<br>…<br>…<br>`</CASE>`<br>`</SWITCH>`<br>`</MESSAGE>`<br>…<br>…<br><br>`</FORM>` |
| **SLEEP** | The tag interrupts script execution for the specified period. The interruption time is obtained from the transferred value multiplied by the time base of 50 ms.<br><br>**Syntax:**<br>`<SLEEP value="Interruption time" />`<br><br>**Example:**<br>Wait time, 1.5 sec.<br>`<SLEEP value="30" />` |
| **STOP** | Interpretation is cancelled at this point. |

| Tag identifier | Meaning |
|---|---|
| **SWITCH** | The **SWITCH** instruction describes a multiple choice. A term is evaluated once and compared with a number of constants. If the expression matches the constants, the instructions are executed within the **CASE** instruction.<br>The **DEFAULT** instruction is executed when none of the constants match the expression.<br><br>**Syntax:**<br>`<SWITCH>`<br>`<CONDITION> Value </CONDITION>`<br>`<CASE value="Constant 1">`<br>`Instructions`<br>`…`<br>`</CASE>`<br>`<CASE value="Constant 2">`<br>`Instructions`<br>`…`<br>`</CASE>`<br>`<DEFAULT>`<br>`Instructions`<br>`…`<br>`</DEFAULT>`<br>`</SWITCH>` |
| **SHOW_CONTROL** | The visibility of a control can be controlled using the tag.<br><br>**Syntax:**<br>`<SHOW_CONTROL name="<name>" type="<type>" />`<br><br>**Attributes:**<br>• **name**<br>Name of the control<br>• **type =** "TRUE" - control becomes visible<br>• **type** = "FALSE" - control becomes invisible (hidden)<br><br>**Example:**<br>`<SHOW_CONTROL name="myEditfield" type="false" />`<br>`…`<br>`…`<br>`<SHOW_CONTROL name="myEditfield" type="true" />` |
| **TYPE_CAST** | The tag converts the data type of a local variable.<br><br>**Syntax:**<br><type_cast name="variable name" type=" new type" /><br>**Attributes:**<br>• **name**<br>Variable name<br>• **type**<br>New data type |
| **THEN** | Operation, if the condition has been fulfilled (**IF, THEN, ELSE**) |

| Tag identifier | Meaning |
|---|---|
| **FOR** | For loop<br>`for (initialization; test; continuation) instruction(s)`<br><br>**Syntax:**<br>`<FOR>`<br>`<INIT>…</INIT>`<br>`<CONDITION>…</CONDITION>`<br>`<INCREMENT>…</INCREMENT>`<br>`Instructions`<br>`…`<br>`</FOR>`<br>The For loop is executed as follows:<br>1. Evaluation of the term **initialization** (INIT).<br>2. Evaluation of the term **test** (CONDITION) as a Boolean term.<br>If the value is false, the For loop is ended.<br>3. Execution of the following instructions.<br>4. Evaluation of the term **continuation** (INCREMENT).<br>5. Continue with 2.<br>All the variables within the INIT, CONDITION, and INCREMENT branches should be created outside the FOR loop.<br><br>**Example:**<br>`<LET name = "count">0</LET>`<br>`<FOR>`<br>`<INIT>`<br>`<OP> count = 0</OP>`<br>`</INIT>`<br>`<CONDITION> count <= 7 </CONDITION>`<br>`<INCREMENT>`<br>`<OP> count = count + 1 </OP>`<br>`</INCREMENT>`<br>`<OP> "plc/qb10" = 1+ count </OP>`<br>`</FOR>` |
| **WAITING** | The tag waits for the component to undergo a hot restart after an NC or drive reset.<br><br>**Attributes:**<br>• **WAITINGFORNC** = "TRUE" - the system waits for the NC to restart<br>• **WAITINGFORDRIVE** = "TRUE" - the system waits for the drives to restart<br><br>**Syntax:**<br>`<WAITING WAITINGFORNC = "TRUE"/>`<br><br>**Example:**<br>`...`<br>`<CONTROL_RESET resetnc = "true" resetdrive = "true"/>`<br>`<WAITING waitingfornc = "true" waitingfordrive = "true" />`<br>`...` |

| Tag identifier | Meaning |
|---|---|
| **WHILE** | WHILE loop<br>`WHILE (Test)`<br>`  Instruction`<br><br>**Syntax:**<br>`<WHILE>`<br>`<CONDITION>…</CONDITION>`<br>`Instructions`<br>`…`<br>`</WHILE>`<br>The While loop executes a sequence of instructions repeatedly while a condition is met. This condition is tested before the sequence of instructions is executed.<br><br>**Example:**<br>`<WHILE>`<br>`<CONDITION> "plc/ib9" == 0 </CONDITION>`<br>`<DATA name = "PLC/qb11"> 15 </DATA>`<br>`</WHILE>` |

| Tag identifier | Meaning |
|---|---|
| **DO_WHILE** | Do while loop<br>`DO`<br>`  Instructions`<br>`WHILE (Test)`<br><br>**Syntax:**<br>`<DO_WHILE>`<br>`Instructions`<br>`…`<br>`<CONDITION>…</CONDITION>`<br>`</DO_WHILE>`<br>The Do while loop comprises an instruction block and a condition. The code within the instruction block is executed first, then the condition is analyzed. If the condition is true, the function executes the code section again. This is continuously repeated until the condition is false.<br><br>**Example:**<br>`<DO_WHILE>`<br>`<DATA name = "PLC/qb11"> 15 </DATA>`<br>`<CONDITIION> "plc/ib9" == 0 </CONDITION>`<br>`</DO_WHILE>` |

| Tag identifier | Meaning |
|---|---|
| **TYPEDEF** | A new identifier for a data type can be specified with the "typedef" tag. This has the benefit for the structure definitions that the data type can be defined once and then used as a data type in a LET instruction.<br>The identifier and type are expected as attributes.<br>The parser supports only the specification of structure definitions.<br><br>Within the type definition, a variable is declared with the "element" tag. The attributes of the tag correspond to the attributes of the let instruction.<br><br>`<typedef name="<identifier>" type="struct">`<br>`  <element name="<Name>" type="<variable type>" />`<br>…<br>…<br>…<br>`</typedef>`<br><br>After defining, the identifier can be used as data type for the LET instruction.<br><br>`<let name="<Variable name>" type="<Identifier>"></let>`<br><br>**Example**:<br>`<typedef name="my_struct" type="struct">`<br>`  <element name="id" type="int" />`<br>`  <element name="name" type="string" />`<br>`  <element name="phone" type="string" />`<br>`</typedef>`<br><br>`<let name="info" type="my_struct"></let>`<br>…<br>…<br>`<op>`<br>`  info.id = 1;`<br>`  info.name = _T"my name";`<br>`  info.phone= _T"0034 45634";`<br>`</op>` |

### 1.6.3 Color coding

The color attribute uses the color coding scheme for the HTML language.

In terms of syntax, color specifications consist of the "#" (hash) character and six digits from the hexadecimal system, with each color represented by two digits.

R – Red

G – Green

B – Blue

**#RRGGBB**

**Example:**

color = "#ff0011"

### 1.6.4 Special XML syntax

Characters with special meanings in XML syntax have to be rewritten if they are to be displayed correctly by a general XML editor.

The following characters are affected:

| Character | Notation in XML |
|---|---|
| < | < |
| > | > |
| & | & |
| " | " |
| ' | ' |

## 1.6.5 Operators

The operation instruction processes the following operators:

| Operator | Meaning |
|---|---|
| = | Assignment |
| == | Equal to |
| <, < | Less than |
| >, > | Greater than |
| <=, <= | Less than or equal to |
| >=, >= | Greater than or equal to |
| \| | OR operation in bits |
| \|\| | Logic OR operation |
| &, & | AND operation in bits |
| &&, && | Logic AND operation |
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| ! | Not |
| != | Not equal to |

Operation instructions are processed from left to right. It may make sense to place terms in parentheses under certain circumstances in order to define the priority for executing subterms.

## 1.6.6 Generating softkey menus and dialog forms

User menus can only be inserted if there is a main-menu tag with the name "main" in the XML description. This tag is called by the system after the <CUSTOM> operating area has been activated. Further menu branches and dialog-box activation can be defined within the tag.

```
<menu name= "MAIN">
<OPEN_FORM name= "main dialogue">
<softkey POSITION="1">
<caption>sub menu 1</caption>
<navigation>sub menu 1</navigation>
</softkey>
<softkey POSITION="8">
<caption>sub menu 8</caption>
<navigation>sub menu 8</navigation>
</softkey>
</menu>

<menu name= "sub menu 1">
<OPEN_FORM name= "dialogue 1">
</menu>

<menu name= "sub menu 8">
<OPEN_FORM name= "dialogue 8“>
</menu>
```

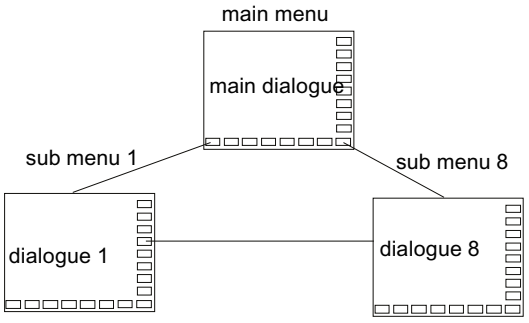
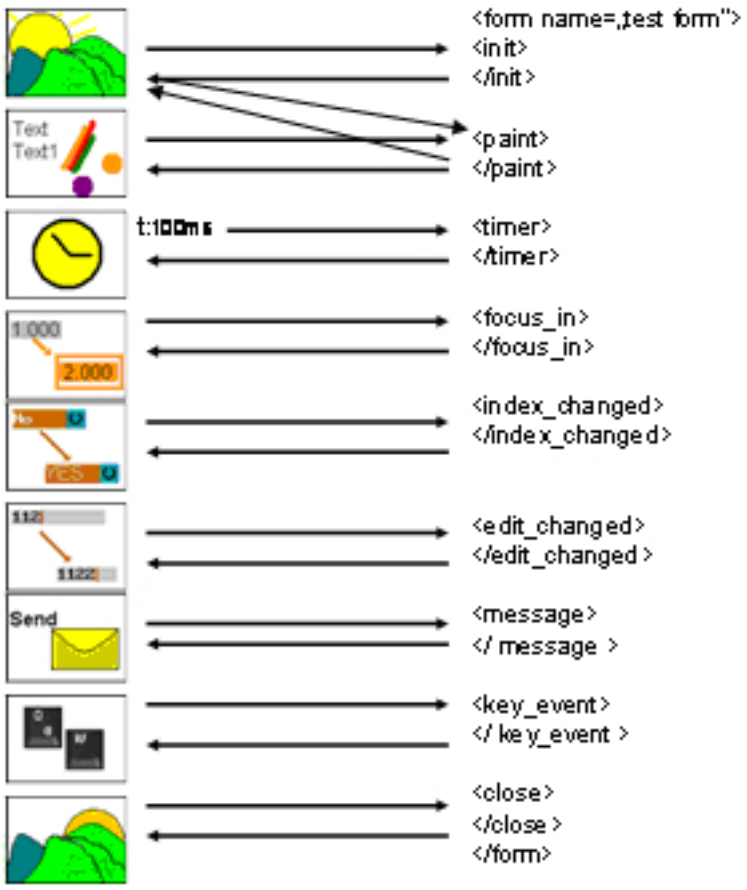



Figure 1-7 Menu structure

| Tag identifier | Meaning |
|---|---|
| **FORM** | This tag contains the description of a user dialog.<br><br>**Attributes:**<br>• **color**<br>Background color of the dialog box (color coding, see Chapter, color coding)<br>• **name**<br>Identifier of the form<br>• **type**<br>*cycle* -attribute specifies a cycle form<br>• **xpos**<br>X-position of the top left corner of the dialog box (optional)<br>• **ypos**<br>Y position of the top left corner (optional)<br>• **width**<br>Extension in the X direction (in pixels) (optional)<br>• **height**<br>Extension in the Y direction (in pixels) (optional) |
| **FORM** continued | **Dialog messages:**<br>• **INIT**<br>• **PAINT**<br>• **TIMER**<br>• **CLOSE**<br>• **FOCUS_IN**<br>• **INDEX_CHANGED**<br>• **EDIT_CHANGED**<br>• **KEY_EVENT**<br>• **MESSAGE** |

| Tag identifier | Meaning |
|---|---|
| **FORM** continued |  |

| Tag identifier | Meaning |
|---|---|
| **FORM** continued | **Syntax:**<br>`<FORM name = "<dialog name>" color = "#ff0000">`<br>**Example:**<br>`<FORM name = "R-Parameter">`<br>` <INIT>`<br>` <DATA_ACCESS type = "true" />`<br>` <CAPTION>R - Parameter</CAPTION>`<br>` <CONTROL name = "edit1" xpos = "322" ypos = "34" refvar = "nck/Channel/Parameter/R[1]" />`<br>` <CONTROL name = "edit2" xpos = "322" ypos = "54" refvar = "nck/Channel/Parameter/R[2]" />`<br>` <CONTROL name = "edit3" xpos = "322" ypos = "74"`<br>` </INIT>`<br><br>`  <PAINT>`<br>` <TEXT xpos = "23" ypos = "34">R - Parameter 1</TEXT>`<br>` <TEXT xpos = "23" ypos = "54">R - Parameter 2</TEXT>`<br>` <TEXT xpos = "23" ypos = "74">R - Parameter 3</TEXT>`<br><br>` </PAINT>`<br><br>`</FORM>` |
| **INIT** | **Dialog box message**<br>The tag is executed immediately after the dialog box is generated. All the input elements and hotlinks for the dialog form should be created here. |

| Tag identifier | Meaning |
|---|---|
| **KEY_EVENT** | Dialog message<br>The tag KEY_EVENT can be integrated in the form to evaluate keyboard events. The system sends the MF2 keyboard code to the active form if the tag is available in a form. If the variable **$actionresult** is not set to zero, the system then subsequently processes the keyboard event.<br>The keyboard code is provided in the variable **$keycode** as an integer value.<br><br>**Example:**<br>The character entered into the variable **exclude_key** should be filtered-out of the input stream.<br><br>`<LET name="stream" type="string"/>`<br>`<LET name="exclude_key" type="string"/>`<br><br>`<FORM name = "keytest_form">`<br>`  <INIT>`<br>`  <CONTROL name = "p1" xpos = "120" ypos = "84" width ="200" refvar="stream" hotlink="true" />`<br>`  <CONTROL name = "p2" xpos = "160" ypos = "104" width ="8" refvar="exclude_key" hotlink="true" />`<br>`  </INIT>`<br><br>`  <PAINT>`<br>`  <text xpos = "8" ypos = "84">data stream</text>`<br>`  <text xpos = "8" ypos = "104">exclude key</text>`<br>`  </PAINT>`<br><br>`  <KEY_EVENT>`<br>`  <LET name="excl_keycode" type="string"/>`<br>`  <OP>excl_keycode = exclude_key</OP>`<br>`  <type_cast name="excl_keycode" type="int" />`<br>`  <PRINT text="%d %d">$keycode, excl_keycode</PRINT>`<br><br>`  <IF>`<br>`  <CONDITION>$keycode == excl_keycode</CONDITION>`<br>`  <THEN>`<br>`  <op> $actionresult = 0</op>`<br>`  </THEN>`<br>`  </IF>`<br><br>`  </KEY_IVENT>`<br><br>`</FORM>` |

| Tag identifier | Meaning |
|---|---|
| MESSAGE | **Dialog message**<br>If the Send_message operation is executed in the script, then the parser processes the tag message. Values P1 and P2 are provided in the variables $message_par1 and $message_par2 (see the "SEND_MESSAGE" tag).<br><br>**Syntax:**<br>`<MESSAGE>`<br>`</MESSAGE>`<br><br>**Example:**<br>`<LET name="user_selection" />`<br><br>`<SOFTKEY POSITION="3">`<br>`<CAPTION>Set%nParameter</CAPTION>`<br>`<SEND_MESSAGE>1, 10</SEND_MESSAGE>`<br>`</SOFTKEY>`<br>…<br>…<br><br>`<FORM>`<br>…<br>…<br>`<MESSAGE>`<br>`<SWITCH>`<br>`<CONDITION>$message_par1</CONDITION>`<br>`<CASE value="1">`<br>`<OP> user_selection = $message_par2 </OP>`<br>…<br>…<br>`</CASE>`<br>`</SWITCH>`<br>`</MESSAGE>`<br>…<br>…<br>`</FORM>` |

| Tag identifier | Meaning |
|---|---|
| **SEND_MESSAGE** | The tag sends a message with two parameters to the active form, which is processed in the tag message (see also MESSAGE).<br><br>**Syntax:**<br>`<SEND_MESSAGE>p1, p2</SEND_MESSAGE>`<br><br>**Example:**<br>`<LET name="user_selection" />`<br><br>`<SOFTKEY POSITION="3">`<br>`<CAPTION>Set%nParameter</CAPTION>`<br>`<SEND_MESSAGE>1, 10</SEND_MESSAGE>`<br>`</SOFTKEY>`<br>…<br>…<br>`<FORM>`<br>…<br>…<br>`<MESSAGE>`<br>`<SWITCH>`<br>`<CONDITION>$message_par1</CONDITION>`<br>`<CASE value="1">`<br>`<OP> user_selection = $message_par2 </OP>`<br>…<br>`</CASE>`<br>`</SWITCH>`<br>`</MESSAGE>`<br>…<br>…<br>`</FORM>` |
| **FOCUS_IN** | **Dialog box message**<br>The tag is called if the system places the focus on a control. In order to identify the control, the system copies the name of the control to variable **$focus_name** and the value of the attribute item_data to variable **$focus_item_data**. The system creates the variables automatically.<br>This message can be used, for example, to output images depending on the focus position.<br>**Example:**<br>`<focus_in>`<br>`<PRINT text="focus on filed:%s, %d">$focus_name,`<br>`$focus_item_data </PRINT>`<br>`</focus_in>` |
| **PAINT** | **Dialog box message**<br>The tag is executed when the dialog box is displayed. All the texts and images which are to be displayed in the dialog box should be specified here.<br>Further, the tag is executed if the system identifies that parts of the dialog box are to be redisplayed. For example, this can be initiated by closing high-level windows. |
| **TIMER** | **Dialog box message**<br>The tag is executed cyclically.<br>Each form is assigned a timer that initiates that the timer - tag is executed approx. every 100 ms. |

| Tag identifier | Meaning |
|---|---|
| **CAPTION** | The tag contains the title of the dialog box.<br>This tag should be used within the INIT tag.<br>The title bar can be divided into multiple columns.<br>To divide the title bar, the "caption" tag must be programmed with the "define_section" attribute before the text is output.<br>The "define_section" attribute specifies the number of columns<br><br>The "property" attribute is used to define the length, starting position, and text alignment for each column. The position and length specification is a percentage value referring to the width of the form.<br>The value of the "value" attribute indicates the column number (beginning with zero)<br><br>`<caption define_sections="3">`<br>`<property value="0" length="20" position="2" alignment="left" />`<br>`<property value="1" length="20" position="79" alignment="right" />`<br>`</caption>`<br><br>The text can then be assigned to the column by additionally specifying the "index" attribute with the column index.<br>`<caption index="0">column1</caption>`<br>`<caption index="1">column2</caption>`<br><br>**Syntax:**<br>`<CAPTION>Titel</CAPTION>`<br><br>**Example:**<br>`<CAPTION>my first dialogue</CAPTION>` |
| **CLOSE** | **Dialog box message**<br>This tag is executed before the dialog box is closed. |
| **CLOSE_FORM** | The tag closes the active dialog.<br>This instruction is only necessary if the dialog is opened by the MMC command and the user is offered a softkey function to close the dialog. Generally, dialogs are automatically managed and do not have to be explicitly closed.<br><br>**Syntax:**<br>`<CLOSE_FORM/>`<br><br>**Example:**<br>`<softkey_ok>`<br>`<caption>OK</caption>`<br>`<CLOSE_FORM />`<br>`<navigation>main_menu</navigation>`<br>`</softkey_ok>` |

| Tag identifier | Meaning |
|---|---|
| **CONTROL** | The tag is used to generate control elements.<br>**Syntax:**<br>`<CONTROL name = "<control name>" xpos = "<X position>" ypos = "<Y position>" refvar = "<NC variable>" hotlink = "true" format = "<format>" />`<br><br>**Attributes:**<br>• **name**<br>Identifier of the field.<br>The identifier simultaneously represents a local variable, and must not be used a multiple number of times in the form.<br>• **xpos**<br>X position of the top left corner<br>• **ypos**<br>Y position of the top left corner<br>• **fieldtype**<br>Field type<br>If no type is specified, the field is set as an edit field.<br>– **edit**<br>Data can be changed<br>– **readonly**<br>Data cannot be changed<br>**combobox**<br>The field displays the corresponding identifiers instead of numerical values.<br>If the field type "**combobox**" is selected, then the expressions to be displayed must also be assigned to the field.<br>The <ITEM> TAG should be used for this purpose.<br>The combo box saves the index of the currently selected text in the variable belonging to the control (see the attribute **refvar**).<br>– **progressbar**<br>A progress bar with a value range of 0 to 100 appears.<br>The valley value and peak value properties can be used to adapt the value range to the data to be displayed. |

| Tag identifier | Meaning |
|---|---|
| **CONTROL** continued | • **fieldtype**<br>– **listbox**<br>The field type generates an empty list box control.<br>Using the tag <ITEM> a list box element can be inserted in the list box.<br>The ITEM attribute **value** allows this element to be assigned a unique value.<br>For example, this can be used to identify the element.<br>Parameters **width** and **height** specify the width and height of the list box.<br>After the control has been created, additional list box elements can be inserted using the functions **AddItem**, **InsertItem** or **LoadItem**.<br>After the control has been created, additional list box elements can be inserted using the function **AddItem** or **InsertItem**. The parameter **itemdata** is not evaluated for this control.<br>– **itemlist**<br>The field type generates a static control, which displays the corresponding identifier instead of numerical values.<br>The <ITEM> tag can be used to assign an identifier to the field.<br>• **item_data**<br>A user-specific integer value can be assigned to the attribute. This value is given as part of the FOCUS_IN message for identifying the focus field.<br>• **refvar**<br>Identifier of the reference variable that can be linked to the field (optional).<br>• **hotlink** = "TRUE" " If the value of the reference variable changes, then the field is automatically updated (optional).<br>• **format**<br>The attribute defines the display format of the specified variable.<br>Formatting data, see print-Tag (optional). |

| Tag identifier | Meaning |
|---|---|
| **CONTROL** continued | **Attributes:**<br>• **time**<br>specifies the data refresh rate (optional).<br>The default value is 200 ms.<br>The following specifications are possible:<br>– **super fast**<br>Refresh time < 100 ms<br>– **fast**<br>Refresh time approx. 100 ms<br>– **normal**<br>Refresh time approx. 200 ms<br>– **slow**<br>Refresh time approx. 500 ms<br>• **color_bk**<br>The attribute sets the background color of the control.<br>• **color_fg**<br>The attribute sets the foreground color of the control.<br>("color coding" see Chapter "Color coding (Page 40)")<br>• **display_format**<br>The attribute defines the processing format of the specified variable. This attribute must be used when accessing a PLC float variable, as the access is realized by reading a double word.<br>The following data formats are permitted:<br>– FLOAT<br>– INT<br>– DOUBLE<br>– STRING<br>Assigning expressions (e.g. text or graphic element to be displayed) to a list box, graphics box or combo box:<br>**Syntax:**<br><ITEM>Expression</ITEM><br><ITEM value ="<Value>">Expression</ITEM> |

| Tag identifier | Meaning |
|---|---|
| **CONTROL** continued | **Example:**<br>`<CONTROL name = "button1" xpos = "10" ypos = "10" fieldtype = " combobox ">`<br>`<ITEM>text1</ITEM>`<br>`<ITEM>text2</ITEM>`<br>`<ITEM>text3</ITEM>`<br>`<ITEM>text4</ITEM>`<br>`</CONTROL>`<br>If any integer value is to be assigned to an expression, the attribute **value = "value"** should be added to the tag.<br>Rather than consecutive numbers, the control variable now contains the item's assigned value.<br><br>**Example:**<br>`<CONTROL name = "button1" xpos = "10" ypos = "10" fieldtype = " combobox ">`<br>`<ITEM value = "10">text1</ITEM>`<br>`<ITEM value = "20">text2</ITEM>`<br>`<ITEM value = "12">text3</ITEM>`<br>`<ITEM value = "1">text4</ITEM>`<br>**</CONTROL>**<br>**Example of a progress bar:**<br>`<CONTROL name = "progress1" xpos = "10" ypos = "10" width = "100" fieldtype = "progressbar" hotlink = "true" refvar = "nck/Channel/GeometricAxis/actProgPos[1]">`<br>`<PROPERTY min = "0" />`<br>`<PROPERTY max = "1000" />`<br>`</CONTROL>`<br>**Example, list box:**<br>`<let name="item_string" type="string"></let>`<br>`<let name="item_data" ></let>`<br><br>`<CONTROL name="listbox1" xpos = "360" ypos="150" width="200" height="200" fieldtype="listbox" />`<br>• Adding elements:<br>Elements are added using the function **additem** or **loaditem**.<br>• Deleting the content:<br>The content is deleted using the function **empty**.<br><br>`<op> item_string = _T"text1\\n" </op>`<br>`<function name="control.additem">_T"listbox1", item_string, item_data </function>`<br>`<op> item_string = _T"text2\\n" </op>`<br>`<function name="control.additem">_T"listbox1", item_string, item_data </function>` |
| **CONTROL** continued | **Example itemlist:**<br>`<CONTROL name = "itemlist1" xpos = "10" ypos = "10" fieldtype = " itemlist">`<br>`<ITEM value = "10">text1</ITEM>`<br>`<ITEM value = "20">text2</ITEM>`<br>`<ITEM value = "12">text3</ITEM>`<br>`<ITEM value = "1">text4</ITEM>`<br>`</CONTROL>` |

| Tag identifier | Meaning |
|---|---|
| **HELP_CONTEXT** | This tag defines the help topic to be called. It should be programmed in the INIT block.<br>The name specified in the attribute is supplemented by the prefix XmlUserDlg_ and is transferred to the help system. The associated structure of the help file should be taken from the topic - generating an online help.<br>**Sequence when activating the help system:**<br>1. Press the "Info" key.<br>2. The dialog supplies the expression "my_dlg_help".<br>3. Parser converts the expression into "XmlUserDlg_my_dlg_help" .<br>4. Activating the help system.<br>5. Submitting the search term "XmlUserDlg_my_dlg_help".<br><br>**Syntax:**<br>`<HELP_CONTEXT name="<context name>" />`<br><br>**Example:**<br>...<br>`<INIT>`<br>...<br>`<CAPTION>my dialogue</CAPTION>`<br>`<HELP_CONTEXT name="my_dlg_help" />`<br>...<br>`</INIT>` |
| **DATA_ACCESS** | The tag controls the behavior of the dialog forms when user inputs are being saved.<br>The behavior should be defined within the INIT tag.<br>If the tag is not used, inputs are buffered in each case.<br>Exception: Controls for which the **hotlink** attribute is set to **true** are always written to and read directly.<br><br>**Attribute::**<br>• **type** = "TRUE" – the input values are not buffered. The dialog form copies the input values to the reference variables directly.<br>• **type =** "FALSE" – the values are only copied to the reference variable with the **UPDATA_DATA** type = "FALSE" tag.<br><br>**Example:**<br>`<DATA_ACCESS type = "true" />` |

| Tag identifier | Meaning |
|---|---|
| **EDIT_CHANGED** | **Dialog box message**<br>This tag is called if the contents of an edit control have changed.<br>To identify the control, the system copies the name of the control into variable **$focus_name** and the value of the attribute **item_data** into variable **$focus_item_data**. The system creates the variables automatically.<br><br>**Example:**<br>`<EDIT_CHANGED>`<br>`<print text="index changed filed:%s, %d"> $focus_name, $focus_item_data </print>`<br>…<br>…<br>`</EDIT_CHANGED>` |
| **INDEX_CHANGED** | **Dialog box message**<br>The tag is called, if the operator changes the selection of a combo box.<br>To identify the control, the system copies the name of the control into the variable<br>**$focus_name** and the value of the attribute **item_data** into the variable **$focus_item_data**. The system creates the variables automatically.<br>**Note:**<br>A reference variable assigned to the control, has not been aligned to the control variable at this point in time and contains the index of the previous selection of the combo box.<br><br>**Example:**<br>`<INDEX_CHANGED>`<br>`<print text="index changed filed:%s, %d"> $focus_name, $focus_item_data </print>`<br>…<br>…<br>`</INDEX_CHANGED>` |
| **MENU** | The tag defines a menu containing the softkey description and the dialog to be opened.<br><br>**Attribute::**<br>• **name**<br>Menu name<br><br>**Syntax:**<br>`<MENU name = "<menu name>">`<br>…<br>`<open_form …>`<br>…<br>`<SOFTKEY …>`<br>`</SOFTKEY>`<br>`</MENU>` |

| Tag identifier | Meaning |
|---|---|
| **NAVIGATION** | This tag defines the menu to be called. This tag can only be set within a softkey block.<br><br>**Syntax:**<br>`<NAVIGATION>menu name</NAVIGATION>`<br><br>**Example:**<br>`<menu name = "main">`<br>`  <softkey POSITION="1">`<br>`  <caption>sec. form</caption>`<br>`  <navigation>sec_menu</navigation>`<br>`  </softkey>`<br>`</menu>`<br><br>`<menu name = "sec_menu">`<br>`  <open_form name = "sec_form" />`<br>`  <softkey_back>`<br>`  <navigation>main</navigation>`<br>`  </softkey_back>`<br>**</menu>** |
| **OPEN_FORM** | The tag opens the dialog form given under the name.<br><br>**Attribute::**<br>• **name**<br>Name of the dialog form<br><br>**Syntax:**<br>`<OPEN_FORM name = "<form name>" />`<br><br>**Example:**<br>`<menu name = "main">`<br>`  <open_form name = "main_form" />`<br>`  <softkey POSITION="1">`<br>`  <caption>main form</caption>`<br>`  <navigation>main</navigation>`<br>`  </softkey>`<br>`</menu>`<br><br>`<form name="main_form">`<br>`  <init>`<br>`  </init>`<br><br>`  <paint>`<br>`  </paint>`<br><br>`</form>` |

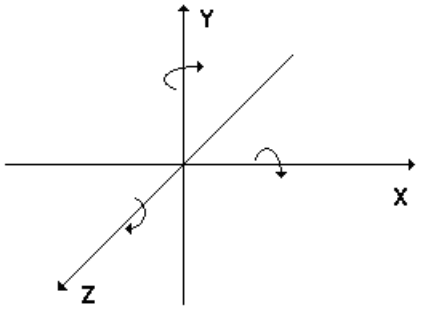
| Tag identifier | Meaning |
|---|---|
| **PROPERTY** | This tag can be used to define additional properties for an operator control.<br><br>**Attributes:**<br>• **max** = "<maximum value>"<br>• **min** = "<minimum value>"<br>• **default** = "<pre-assignment>"<br>• **factor** = "conversion factor"<br>• **color_bk** = "<background color coding>"<br>• **color_fg** = "<font color coding>"<br>• **password** = "<true>" - entered character is displayed with "*"<br>• **multiline** = "<true>" - permits multi-line inputs in an edit control<br>• **disable** = "<true/false>" - locks/permits the input in an edit control<br>• **transparent**= "Transparent color of a bitmap"<br>Color coding (for details on color coding, see chapter, Color coding)<br>• **tooltip** = information text is displayed if the cursor is set to the control.<br>The syntax is: <property tooltip="information text" /><br>**Example:**<br>`<CONTROL name = "progress1" xpos = "10" ypos = "10" width = "100" fieldtype = "progressbar" hotlink = "true" refvar = "nck/Channel/GeometricAxis/actProgPos[1]">`<br>`<PROPERTY min = "0" />`<br>`<PROPERTY max = "1000" />`<br>`</CONTROL>`<br>`<CONTROL name = "edit1" xpos = "10" ypos = "10">`<br>`<PROPERTY min = "20" />`<br>`<PROPERTY max = "40" />`<br>`<PROPERTY default = "25" />`<br>`</CONTROL>` |

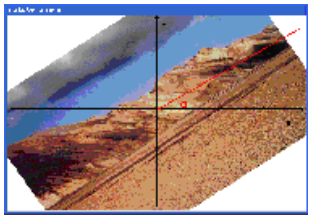
| Tag identifier | Meaning |
|---|---|
| **SOFTKEY** | The tag defines the properties and responses of a softkey.<br>**Attributes:**<br>• **position**<br>Number of the softkey. 1-8 horizontal softkeys, 9-16 vertical softkeys<br>The following attributes become effective from:<br>• **type**<br>Defines the property of the softkey.<br>user_controled - The script defines how the softkey is displayed<br>toggle_softkey - The softkey is displayed alternating between pressed and not pressed<br>• **refvar**<br>Should only be used in conjunction with toggle_softkey .<br>Reference variable, into which the actual softkey property is copied.<br>A variable, type "String" should be specified, which includes the properties pressed, not pressed or locked (see tag state).<br>• **picture**<br>Using the attribute, a bitmap can be output left justified on the softkey. The complete path name should be specified.<br>The number of text characters that can be displayed is reduced to the width of the bitmap.<br><br>The following additional actions can be defined within the softkey block:<br>• **picturealignment**<br>The image orientation is specified by this attribute.<br>The image is aligned with the left side of the softkey by default.<br>The following values can be specified for alignment:<br>– **top**<br>Top edge<br>– **bottom**<br>Bottom edge<br>– **left**<br>Left edge<br>– **right**<br>Right edge<br>– **center**<br>Centered<br>• **caption**<br>Softkey text<br>• **state**<br>Should only be used in conjunction with user_controlled .<br>The tag assigns the required softkey display to the system.<br>**Syntax:**<br><state type="<state>" /><br>The following strings can be specified:<br>– **notpressed**<br>The softkey is displayed as being not pressed.<br>– **pressed**<br>The softkey is displayed as being pressed. |

| Tag identifier | Meaning |
|---|---|
| | – **disabled**<br>The softkey is locked and is displayed in gray.<br>• **navigation**<br>• **update_controls**<br>• **function** |
| **SOFTKEY** continued | **Syntax:**<br>Standard softkey:<br><state type="<softkey state>" /><br><softkey position = "<1>"><br>…<br>…<br></softkey><br>or<br>Script-controlled softkey:<br><softkey position = "<1>" type="<user_defined>" ><br><state type="<softkey state>" /><br>…<br>…<br></softkey><br><br>or<br>Toggle softkey:<br><softkey position = "<1>" type="<toggle_softkey>" refvar="<variable name>" ><br>…<br>…<br></softkey><br><br>**Example:**<br><br><let name="define_sk_type" type="string">PRESSED</let><br><let name="sk_type">1</let><br><br><softkey POSITION="1" type="user_controled" ><br><caption>Toggle%nSK</caption><br><if><br><condition>sk_type == 0 </condition><br><then><br><op> sk_type = 1 </op><br><op> define_sk_type = _T"PRESSED" </op><br></then><br><else><br><op> define_sk_type = _T"NOTPRESSED" </op><br><op> sk_type = 0 </op><br></else><br></if><br><state type="$$$define_sk_type" /><br>…<br>…<br><br></softkey> |

| Tag identifier | Meaning |
|---|---|
| **SOFTKEY** continued | **Example:**<br>or<br><br><let name="curr_softkey_state" type="string">PRESSED</let><br></softkey><br><br><softkey POSITION="3" type="toggle_softkey" refvar="curr_softkey_state"><br><caption>Toggle%nSK</caption><br>…<br><br></softkey><br><br><br><br><br><br> |
| **SOFTKEY_OK** | The tag defines the response of the softkey "OK".<br><br><br><br><br>The following additional actions can be defined within the softkey block:<br>• **navigation**<br>• **update_controls**<br>• **function**<br><br>**Syntax:**<br><SOFTKEY_OK><br>…<br>…<br></SOFTKEY_OK> |
| **SOFTKEY_CANCEL** | The tag defines the response of the softkey "Cancel".<br><br><br><br><br>The following additional actions can be defined within the softkey block:<br>• **navigation**<br>• **update_controls**<br>• **function**<br><br>**Syntax:**<br><SOFTKEY_CANCEL><br>…<br>…<br></SOFTKEY_CANCEL> |

| Tag identifier | Meaning |
|---|---|
| **SOFTKEY_BACK** | The tag defines the response of the softkey "Back".<br><br><br>The following additional actions can be defined within the softkey block:<br>• **navigation**<br>• **update_controls**<br>• **function**<br>**Syntax:**<br>`<SOFTKEY_BACK>`<br>...<br>...<br>`</SOFTKEY_BACK>` |
| **SOFTKEY_ACCEPT** | The tag defines the response of the softkey "Accept".<br><br><br>The following additional actions can be defined within the softkey block:<br>• **navigation**<br>• **update_controls**<br>• **function**<br>**Syntax:**<br>`<SOFTKEY_ACCEPT>`<br>...<br>...<br>`</SOFTKEY_ACCEPT>` |
| **TEXT** | The tag is used to display a text in the specified position.<br>If an alarm number is used, the dialog box displays the text which is saved for the number.<br>**Syntax:**<br>`<TEXT xpos = "<X position>" ypos = "<Y position>"> Text </TEXT>`<br>**Attributes:**<br>• **xpos**<br>X position of the top left corner<br>• **ypos**<br>Y position of the top left corner<br>• **color**<br>Text color (color coding, see Chapter Color coding (Page 40))<br>**Value:**<br>Text to be displayed |

| Tag identifier | Meaning |
|---|---|
| **IMG** | The tag is used to display an image in the specified position. The BMP and PNG image formats are supported.<br><br>**Syntax:**<br>`<IMG xpos = "<X position>" ypos = "<Y position>" name = "<name>" />`<br>`<IMG … xrot="angle X axis" yrot=" angle Y axis " zrot=" angle Z axis " />`<br><br>**Attributes:**<br>• **xpos**<br>X position of the top left corner<br>• **ypos**<br>Y position of the top left corner<br>• **name**<br>complete path name<br>• **transparent**<br>Transparent color of the bitmap (see Chapter "Color coding"):<br>• **xrot, yrot, zrot** (only for 802Dsl/808D)<br>In the axes X, Y and Z, the image rotates around a point of rotation.<br>• **xref, yref**, **zref** (only for 802Dsl / 808D)<br>Define point of rotation<br>If a point of rotation is not specified, then the image rotates around the image center point. In this case, the Z plane is 0.<br><br> |

| Tag identifier | Meaning |
|---|---|
| **IMG** Continued | **Example**:<br>The image is rotated through 34 degrees around the Z axis:<br>`<img xpos = "5" ypos = "23" name = "f:/appl/pic3.bmp" height="355" width="550" zrot="34" color_bk="#ffffff" ></img>`<br><br>**Optional:**<br>If the image display is to differ from the original size, the dimensions can be defined using the attributes **width** and **height**.<br>• **width**<br>Width in pixels<br>• **height**<br>Height in pixels<br>**Examples:**<br>`<IMG xpos = "20" ypos = "40" name = "f:/appl/test.bmp" />`<br>`<IMG xpos = "5" ypos = "23" name = "f:/appl/test.bmp" height = "355" width = "550"/>` |
| **BOX** | The tag draws a rectangle at the specified position, colored as indicated.<br>**Syntax:**<br>`<BOX xpos = "<X position>" ypos = "<Y position>" width = "<X extension>" height = "<Y extension>" color = "<Color code>" />`<br>**Attributes:**<br>• **xpos**<br>X position of the top left corner<br>• **ypos**<br>Y position of the top left corner<br>• **width**<br>Extension in X direction (in pixels)<br>• **height**<br>Extension in Y direction (in pixels)<br>• **color**<br>Color coding (for details on color coding, see chapter, Color coding) |

| Tag identifier | Meaning |
|---|---|
| **FUNCTION** | **Function call**<br>The tag executes the function body, which is specified under the attribute "**name**.<br><br>**Attributes:**<br>• **name** = "Name of the function body"<br>• **return** = "Variable name for saving the result of the function"<br><br>**Values:**<br>List of variables to be transferred to the function body. The variables must be separated by a comma. A maximum of 10 parameters can be transferred.<br>It is also possible to specify constants or text expressions as call parameters. The identifier _T should be placed at the start as a means of identifying text terms.<br><br>**Syntax:**<br>`<FUNCTION name = "<function name>" />`<br>Calling function expects a return value<br>`<FUNCTION name = "<function name>" return = "<Variablenname>" />`<br>Parameter transfer<br>`<FUNCTION name = "<function name>"> var1, var2, var3 </FUNCTION>`<br>`<FUNCTION name = "<function name>"> _T"Text", 1.0, 1 </FUNCTION>`<br><br>**Examples:**<br>See "FUNCTION_BODY". |

| Tag identifier | Meaning |
|---|---|
| FUNCTION_BODY | **Function body**<br>The tag contains the function body of a subfunction. The function body needs to be programmed within the DialogGui tag.<br><br>**Attributes:**<br>• **name** = "Name of the function body"<br>• **parameter** = "Parameter list" (optional)<br>The attribute lists the transfer parameters that are required. The parameters must be separated by a comma.<br>When the function body is called, the values of the parameters specified in the function call are copied to the transfer parameters listed.<br>• **return** = "true" (optional)<br>If the attribute is set to true then the local variable $return is created. The function's return value which is forwarded to the calling function on quitting the function should be copied to this variable.<br><br>**Syntax:**<br><br>**Function body without parameter**<br>`<FUNCTION_BODY name = "<function name>">`<br>…<br>…<br>…<br>`</ FUNCTION_BODY>`<br><br>**Function body with parameter**<br>`<FUNCTION_BODY name = "<function_name>" parameter = "<p1, p2, p3>">`<br>…<br>`<LET name = "tmp></LET>`<br>`<OP> tmp = p1 </OP>`<br>…<br>`</FUNCTION_BODY>`<br><br>**Function body with return value**<br>`<FUNCTION_BODY name = "<function_name>" parameter = "<p1, p2, p3>" return = "true">`<br>…<br>`<LET name = "tmp></LET>`<br>`<OP> tmp = p1 </OP>`<br>…<br>`<OP> $return = tmp </OP>`<br>`</FUNCTION_BODY>` |

| Tag identifier | Meaning |
|---|---|
| **FUNCTION_BODY** continued | **Example:**<br>`<function_body name = "test" parameter = "c1,c2,c3" return = "true">`<br>`<LET name = "tmp">0</LET>`<br>`<OP> tmp = c1+c2+c3 </OP>`<br>`<OP> $return = tmp </OP>`<br>`</function_body>`<br>…<br>…<br>…<br>`<LET name = "my_var"> 4 </LET>`<br>`<function name = "test" return = " my_var "> 2, 3,4</function>`<br>`<print text = "result = %d"> my_var </print>`<br>…<br>… |

| Tag identifier | Meaning |
|---|---|
| **REQUEST** | The tag is used to add a variable to the cyclic reading service (hotlink). As a consequence, the access time to variables, which are not linked to the control, is reduced.<br>If a function is to be called automatically when a value changes, then the name of the function should be specified as an additional attribute.<br>This tag is only processed within the INIT operation.<br><br>**Attributes:**<br>• **name**<br>Address identifier<br>• **function**<br>Function name<br><br>**Syntax:**<br>`<REQUEST name = "<NC-Variable>" />`<br>or<br>`<REQUEST name = "<NC-Variable>" function="<function name>"/>`<br><br>**Example:**<br>`<request name ="plc/mb10" />`<br><br>or<br><br>`<function_body name="my_function" >`<br>`<print text="value changed" />`<br>`</function_body>`<br>…<br>…<br>…<br>`<request name ="plc/mb10" function="my_function"/>` |
| **UPDATE_CONTROLS** | The tag runs a comparison between the operator controls and the reference variables.<br><br>**Attribute::**<br>• **type**<br>The attribute defines the direction of the data comparison.<br>**= TRUE** – data is read from the reference variables and copied to the operator controls.<br>**= FALSE** – Data is copied from the operator controls to the reference variables.<br><br>**Syntax:**<br>`<UPDATE_CONTROLS type = "<Direction>"/>`<br><br>**Example:**<br>`<SOFTKEY_OK>`<br>`< UPDATE_CONTROLS type="false"/>`<br>`</SOFTKEY_OK>` |

# 1.7 Generating user menus

## 1.7.1 Creating processing cycle forms

The cycle support function allows the automatic creation and decompilation of a cycle call through the form dialog.

To manage this functionality, the following tags are available:

- NC_INSTRUCTION
- CREATE_CYCLE

To mark a cycle form, in the FORM tag, the attribute type should be specified with the value cycle. This marking allows the NC_INSTRUCTION to be processed.

### Example

```
<FORM name = "cycle100_form" type= "CYCLE">
…
…
</FORM>
```

The NC_INSTRUCTION tag contains the cycle call to be generated. All cycle parameters should be reserved using space retainers.

### Example

```
<FORM name = "cycle100_form" type= "CYCLE">
<NC_INSTRUCTION refvar= "cyc_string" >Cycle100 ($p1, $p2, $p3)</
NC_INSTRUCTION>
…
…
…
</FORM
```

The CREATE_CYCLE tag prepares the values saved in the space retainer variables and generates the NC instruction.

This is then copied to the specified variable.

| Tag identifier | Description |
|---|---|
| **NC_INSTRUCTION** | This tag is used to define the NC instruction to be generated.<br>All listed cycle parameters are automatically created as string variables of the **FORM** and are available to the **FORM**.<br><br>Precondition: The FORMattribute **type** is set to the value **CYCLE**.<br><br>**Attribute:**<br>• **refvar**<br>If the tag is assigned a reference variable, all parameters are pre-assigned with the values from the NC block saved in the reference variables.<br>**Syntax:**<br>`<NC_INSTRUCTION> NC instruction with space retainers </NC_INSTRUCTION>`<br><br>**Example:**<br>`<let name="cyc_string" type="string"> Cycle100(0, 1000, 5)</let>`<br>...<br>...<br>...<br>`<FORM name = "cycle100_form" type= "CYCLE">`<br>`<NC_INSTRUCTION refvar= "cyc_string" >Cycle100($p1, $p2, $p3)</ NC_INSTRUCTION>`<br>..<br>…<br>…<br>`</FORM>` |

| Tag identifier | Description |
|---|---|
| **CREATE_CYCLE** | The tag generates an NC block, whose syntax is defined by the value of the **NC_INSTRUCTION** tag.<br>Before generating the NC instruction, the parser calls the CYCLE_CREATE_EVENT tag of the FORM. This tag can be used to calculate the cycle parameters.<br>**Syntax:**<br>`<CREATE_CYCLE/>`<br><br>**Attribute:**<br>• **refvar**<br>If the tag is assigned a reference variable, the NC instruction is copied to this variable.<br><br>**Example:**<br>`<LET name="cyc_string" type="string"> Cycle100(0, 1000, 5)</LET>`<br><br>`<SOFTKEY_OK>`<br>`  <caption>OK</caption>`<br>`  <CREATE_CYCLE />`<br>`  <close_form />`<br>`  <navigation>main_menu</navigation>`<br>`</SOFTKEY_OK>`<br><br>`or`<br><br>`<SOFTKEY_OK>`<br>`  <caption>OK</caption>`<br>`  <CREATE_CYCLE refvar= "cyc_string" />`<br>`  <close_form />`<br>`  <navigation>main_menu</navigation>`<br>`</SOFTKEY_OK>` |

### 1.7.2 Substitution characters

The system offers the option of defining control properties (attribute values) for the runtime. In order to use this function, the desired property must be set in a local variable and the variable name must be transferred to the tag as an attribute value preceded by the **character $**.

If the tag expects a string as attribute value or value, the $$$ characters must be placed in front of the variable name.

**Example:**

```
<let name="my_ypos">100</let>
<let name="field_name" type="string"></let>

<control name = "edit1" xpos = "322" ypos = "$my_ypos" refvar="nck/
Channel/Parameter/R[1]" />

<op>my_ypos = my_ypos +20 </op>

<control name = "edit2" xpos = "322" ypos = "$my_ypos" refvar="nck/
Channel/Parameter/R[2]" />

<print name =" field_name" text="edit%d">3</print>
<op>my_ypos = my_ypos +20 </op>

<control name = "$field_name" xpos = "322" ypos = "$my_ypos"
refvar="nck/Channel/Parameter/R3]" />

<caption>$$$field_name</caption>
```

## 1.8 Addressing components

Address identifiers for the desired data must be created to address NC variables, PLC blocks or drive data. An address consists of the subpaths **component name** and **variable address**. A slash should be used as a separating character.

### 1.8.1 PLC addressing

Addressing the PLC starts with the path section **plc**.

Table 1-3 The following addresses are permissible:

| | |
|---|---|
| **DBx.DB(f)** | Data block |
| **I(f)x** | Input |
| **Q(f)x** | Output |
| **M(f)x** | Bit memory |
| **V(f)x** | Variable |

| | |
|---|---|
| **DBx.DBXx.b** | Data block |
| **Ix.b** | Input |
| **Qx.b** | Output |
| **Mx.b** | Bit memory |
| **Vx.b** | Variable |

Table 1-4 Data format **f**:

| | |
|---|---|
| B | Byte |
| W | Word |
| D | Double word |

Data format identification is not applicable to bit addressing.

Address **x**:

Valid S7-200 address identifier

Bit addressing:

**b** – Bit number

**Examples:**

```
<data name = "plc/mb170">1</data>
<data name = "i0.1"> 1 </data>
<op> "m19.2" = 1 </op>
```

## 1.8.2 Addressing NC variables

Addressing the NC variables starts with the path section **nck**.

This section is followed by the data address; its structure should be taken from the SINUMERIK 802D sl Parameter Manual.

**Example:**

```
<LET name = "tempStatus"></LET>
<OP> tempStatus ="nck/channel/state/chanstatus" </OP>
```

## 1.8.3 Generating NC/PLC addresses during the runtime

It is possible to generate an address identifier during runtime.

In this case, the content of a string variable is used as address in an operation statement as well as in the nc.cap.read and nc.cap.write functions.

Observe the following for this type of addressing mode:

- Write the variable names in quotation marks.
- Use three ‚$' characters as prefix for variable names.

**Syntax:**

```
"$$$variable name"
```

**Example:**

```
<LET name="var_adr" type="string"></LET>
<PRINT name="var_adr" text="VB9000%d"> 2000</PRINT>
```

or

```
<PRINT name="var_adr" text="DB9000.DBW%d"> 2000</PRINT>
<OP> "$$$var_adr" = 1 </OP>
```

## 1.8.4 Addressing drive components

Addressing the drive components starts with the path section **drive**.

Then the drive device is specified:

**CU**

**DC**

The parameter to be set is added to this section.

**Example:**

```
<LET name="r0002_content"></LET>
<LET name="p107_content"></LET>

<!— Reading of value r0002 on the CU ->
```

```
<OP> r0002_content = "drive/cu/r0002" </OP>
<OP> r0002_content = "drive/cu/r0002[CU1]" </OP>

<!— Reading of value r0002 on NX1 ->
<OP> r0002_content = "drive/cu/r0002[CU2]" </OP>

<!— Reading of value p107[0] on the CU ->
<OP> p107_content = "drive/cu/p107[0]" </OP>
<PRINT text="%d"> p107_content </PRINT>

<!— Reading of value p107[0] on the CU ->

<OP> p107_content = "drive/cu/p107[0, CU1]" </OP>
<PRINT text="%d"> p107_content </PRINT>

<!— Reading of value p107[0] on the NX1 ->

<OP> p107_content = "drive/cu/p107[0, CU2]" </OP>
<PRINT text="%d"> p107_content </PRINT>
```

**Addressing the drive objects:**

To address individual objects, the desired object should be entered in square brackets after the parameter.

**Note**

The drive object number differs from the numbering used in the drive dialog, since the CU components, ALM, and all connected hubs are integrated with the continuous numbering.

The DO number can be established as follows:

- All connected drive objects are listed in field p978 of the relevant CU corresponding to their slot number.
- It is a question of establishing the field index for the desired slot and adding one to this number. This value is the DO index needed for addressing purposes.

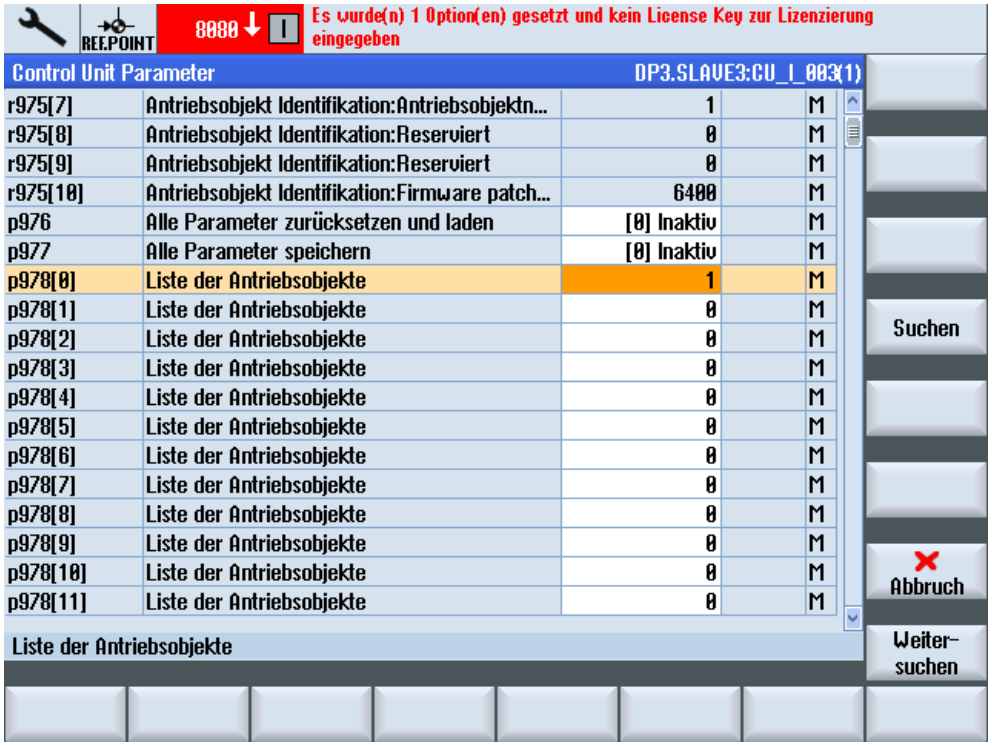



Figure 1-8 Drive parameter p0978 [ ...] at the control

```
Parameter number[do<DO-index>]
```

**Example:**

```
p0092[do1]
```

Alternatively, the drive index can be read from a local variable using **$<variable name>** "substitution characters".

```
z.B. DO$local variable
```

**Example:**

```
<DATA name ="drive/cu/p0092">1</DATA>
<DATA name ="drive/dc/p0092[do1] ">1</DATA>
```

**Indirect addressing:**

```
<LET name = "driveIndex> 0 </LET>
<OP> driveIndex = $ctrlout_module_nr[0, AX1] </OP>
<DATA name ="drive/dc[do$driveIndex]/p0092">1</DATA>
```

## 1.8.5 Addressing machine and setting data

Drive and setting data is identified by the character $ followed by the name of the data.

**Machine data:**

$Mx_<name[**index, AX<axis_number>**]>

**Setting data:**

$Sx_<name[index, AX<axis_number>]>

**x:**

N – General machine or setting data

C – Channel-specific machine or setting data

A – Axis-specific machine or setting data

**Index:**

For a field, the parameter indicates the index of the data.

**AX<axis_number>:**

The required axis (**<axis_number>**) has to be specified for axis-specific data.

Alternatively, the axis index can be read from a local variable using **$<variable name>** "substitution characters".

e.g. AX$localvariable

**Example:**

```
<DATA name ="$MN_AXCONF_MACHAX_NAME_TAB[0] ">X1</DATA>
```

**Direct addressing of the axis:**

```
<DATA name ="$MA_CTRLOUT_MODULE_NR[0, AX1] ">1</DATA>
…
…
```

**Indirect addressing of the axis:**

```
<LET name ="axisIndex"> 1 </LET>
<DATA name ="$MA_CTRLOUT_MODULE_NR[0, AX$axisIndex] ">1</DATA>
```

## 1.8.6 Addressing user data

Addressing user data starts with the path section **gud**, followed by the GUD name.

For a field, after the name, the required field index should be specified in square brackets.

**Example:**

```
<DATA name ="gud/syg_rm[0]"
<OP>"gud/syg_rm[0]" 0 10 </op>
```

### Addressing the global user data

Addressing starts with the path section gud, followed by the specification of the area CHANNEL. This address section is followed by the specification of the GUD areas:

| GUD areas | Assignment |
|---|---|
| sgud | Siemens GUD |
| mgud | Machine manufacturer GUD |
| ugud | User GUD |

Then enter the GUD name. If an array is to be addressed, the name is followed by the array subscript in square brackets.

**Example:**

```
<data name ="gud/channel/mgud/syg_rm[0]">1</data>
<op>"gud/channel/mgud/syg_rm[0]" = 5*2 </op>
```

# 1.9 Predefined functions

The script language offers various string processing and standard mathematical functions.

The function names listed below are reserved and cannot be overloaded.

| Function name | Description |
|---|---|
| **Ncfunc cap read** | The function copies a value from the specified address into a local variable. If the read operation was error-free, then the return variable contains the value zero.<br><br>Contrary to the operation instruction, in the event of a fault, this function does not interrupt the processing of the script operations.<br><br>**Syntax:**<br>`<function name="ncfunc.cap.read" return="error">`<br>`lokale variable, "address"</function>`<br><br>**Example:**<br>`<let name="error"></let>`<br>`<function name="ncfunc.cap.read" return="error">`<br>`3, "drive/cu/p0009"</function>`<br>`<if>`<br>`<condition>error != 0</condition>`<br>`<then>`<br>`<break />`<br>`</then>`<br>`</if>` |
| **Ncfunc cap write** | The function writes a value into the specified variable. If the write operation was error-free, then the return variable contains the value zero.<br><br>Contrary to the operation instruction, in the event of a fault, this function does not interrupt the processing of the script operations.<br><br>**Syntax:**<br>`<function name="ncfunc.cap.read" return="error">`<br>`local variable or constant, "address"</function>`<br><br>**Example:**<br>`<let name="error"></let>`<br>`<function name="ncfunc.cap.write" return="error">`<br>`0, "drive/cu/p0009"</function>`<br>`<if>`<br>`<condition>error != 0</condition>`<br>`<then>`<br>`<break />`<br>`</then>`<br>`</if>` |

| Function name | Description |
|---|---|
| **Ncfunc PI-Service** | Jobs can be transferred to the NCK using the program invocation (PI) service.<br>If the service has been executed error-free, the function returns the value 1 in the return variable.<br><br>**Manipulation of the tool list**<br><br>_N_CREATO - Create tool<br>_N_DELETO - Delete tool<br>_N_CREACE - Create tool cutting edge<br>_N_DELECE - Delete tool cutting edge<br><br>**Activation of work offsets**<br><br>_N_SETUFR - Activates the actual user frame<br>_N_SETUDT - Activates the actual user data<br><br>**Block search**<br><br>_N_FINDBL - Activate block search<br>_N_FINDAB - Cancel block search<br><br>**Syntax:**<br>`<function name="ncfunc.pi_service" return="return var"> pi name, var1, var2, var3, var4, var5 </ function>`<br><br>**Attributes:**<br>• name - function name<br>• return- Name of the variable in which the execution result is saved<br>– Value == 1 – job executed successfully<br>– Value == 0 – faulty job<br><br>**Tag values:**<br>• pi name - Name of the PI service (string)<br>• var1 tovar5 - PI specific arguments |

| Function name | Description |
|---|---|
| **Ncfunc PI-Service** Continued | **Arguments:**<br>• _N_CREATO<br>var1 - Tool number<br>• _N_DELETO<br>var1 - Tool number<br>• _N_CREACE<br>var1 - Tool number<br>var2 - Cutting edge number<br>• _N_DELECE<br>var1 - Tool number<br>var2 - Cutting edge number<br>• _N_SETUFR<br>No arguments<br>• _N_SETUDT<br>var1- User data area to be activated<br>– 1 - Tool offset data<br>– 2 - Active basic frame<br>– 3 - Active adjustable frame<br>• _N_FINDBL<br>var1 - Search mode<br>– 2 - Search with contour calculation<br>– 4 - Search for the block end point<br>– 1 - Block search without calculation.<br>• _N_FINDAB<br>No arguments<br><br>**Example:**<br>• Creating a tool – tool number 3<br><br>`<function name="ncfunc.pi_service">_T"_N_CREATO", 3</function>`<br><br>• Delete cutting edge 1 of tool 5<br><br><function name="ncfunc.pi_service">_T"_N_DELECE", 5, 1</function> |

| Function name | Description |
|---|---|
| **Ncfunc display resolution** | This function supplies the conversion rule for floating point numbers defined in the control. A string variable must be provided as variable.<br><br>See also display machine data MD203 DISPLAY_RESOLUTION and MD204 DISPLAY_RESOLUTION_INCH<br><br>**Syntax:**<br>`<function name="ncfunc.displayresolution" return="dislay_res" />`<br><br>**Example:**<br>`<let name="dislay_res" type="string"></let>`<br>`...`<br>`<function name="ncfunc.displayresolution" return="dislay_res" />`<br><br>`<control name = "cdistToGo" xpos = "210" ypos = "156" refvar="nck/Channel/GeometricAxis/ progDistToGo[2]" hotlink="true" height="34" fieldtype="readonly" format="$$$dislay_res" time="superfast" color_bk="#ffffff"/>` |
| **Ncfunc bico to int** | The function converts a string specified in the BICO format into an integer value. (see SINAMICS).<br><br>**Syntax:**<br>`<function name="ncfunc.bicotoint" return="integer variable">bico-string</function>`<br><br>**Example:**<br>`<let name="s_np0480_0" type="string"></let>`<br>`<let name="i_p0480_0">0</let>`<br><br>`<function name="ncfunc.bicotoint" return="i_p0480_0">s_np0480_0</function>` |
| **Ncfunc int to bico** | The function converts an integer value into a BICO format string. (see SINAMICS).<br><br>**Syntax:**<br>`<function name="ncfunc.inttobico" return="string variable">integer variable</function>`<br><br>**Example:**<br>`<function name="ncfunc.inttobico" return="s_p0480_0">"drive/dc/p0480[0, DO2]"</ function>` |

| Function name | Description |
|---|---|
| **Ncfunc is bico str valid** | This function returns the value zero if it involves a string specified in the BICO format. (see SINAMICS)<br><br>**Syntax:**<br>`<function name="ncfunc.isbicostrvalid" return="integer variable">string varaible</ function>`<br><br>**Example:**<br>`…`<br>`<let name="s_np0480_0" type="string"></let>`<br>`…`<br>`…`<br>`<control name = "cp0480_0" xpos = "402" ypos = "76"  hotlink="true" refvar="s_np0480_0" >`<br>`<property item_data="4001" />`<br>`</control>`<br>`…`<br>`…`<br>`<function name="ncfunc.isbicostrvalid" return="valid">cp0480_0</function>` |
| **Ncfunc password** | This function sets or deletes a password level.<br>• Set password:<br>The password should be specified for the required password level as parameter.<br>• Delete password:<br>A blank string deletes the password level.<br><br>**Syntax:**<br>`<function name="ncfunc.password">password </function>`<br><br>**Example:**<br>`<let name="password" type="string"></let>`<br><br>`<function name="ncfunc.password" > password </ function>`<br>`<function name="ncfunc.password" > _T"CUSTOMER" </ function>`<br><br>Delete password:<br><function name="ncfunc.password" > _T"" </function> |

| Function name | Description |
|---|---|
| **Control form color** | This function provides the text or background color of the dialog box.<br><br>**Range:**<br>• BACKGROUND – request color value of the background<br>• TEXT – request color value of the text (foreground)<br><br>**Syntax:**<br><function name="control.formcolor" return="variable">_T"range"</function><br><br>**Example:**<br><let name="bk_color"></let><br><br><function name="control.formcolor" return="bk_color">_T"BACKGROUND"</function> |
| **Control local time** | The function copies the local time in a field with 7 array elements.<br>The name of the variable is expected as call parameter.<br>The following is stored in an array element:<br>• Index 0 - year<br>• Index 1 - month<br>• Index 2 - weekday<br>• Index 3 - day<br>• Index 4 - hour<br>• Index 5 - minute<br>• Index 6 - second<br><br>**Syntax:**<br><function name="control.localtime">_T"time_array"</function><br><br>**Example:**<br><!-- index<br>0 = Year<br>1 = Month<br>2 = Day of week<br>3 = Day<br>4 = Hour<br>5 = Minute<br>6 = Second<br>--><br><let name="time_array" dim="7" /><br><br><function name="control.localtime">_T"time_array"</function> |

| Function name | Description |
|---|---|
| **String to compare** | Two strings are compared with one another from a lexicographical perspective.<br>The function gives a return value of zero if the strings are the same, a value less than zero if the first string is smaller than the second string or a value greater than zero if the second string is smaller then the first string.<br>**Parameter:**<br>**str1** - string<br>**str2** - comparison string<br><br>**Syntax:**<br>`<function name="string.cmp" return ="<int var>" > str1, str2 </function>`<br><br>**Example:**<br>`<let name="rval">0</let>`<br>`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`<br>`<let name="str2" type="string">A brown bear hunts a brown dog.</let>`<br><br>`<function name="string.cmp" return="rval"> str1, str2 </function>`<br>**Result:**<br>rval= 0 |

| Function name | Description |
|---|---|
| **String to compare without making a distinction between uppercase/lowercase** | Two strings are compared from a lexicographical perspective (the comparison is not case-sensitive).<br>The function gives a return value of zero if the strings are the same, a value less than zero if the first string is smaller than the second string or a value greater than zero if the second string is smaller then the first string.<br><br>**Parameter:**<br>**str1** - string<br>**str2** - comparison string<br><br>**Syntax:**<br>`<function name="string.icmp" return ="<int var>" > str1, str2 </function>`<br><br>**Example:**<br>`<let name="rval">0</let>`<br><br>`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`<br>`<let name="str2" type="string">A brown Bear hunts a brown Dog.</let>`<br>`<function name="string. icmp" return="rval"> str1, str2 </function>`<br>**Result:**<br>rval= 0 |
| **String left** | The function extracts the first nCount character from string 1 and copies this to the return variable.<br><br>**Parameter:**<br>**str1** - string<br>**nCount** - number of characters<br><br>**Syntax:**<br>`<function name="string.left" return="<result string>"> str1, nCount </function>`<br><br>**Example:**<br>`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`<br>`<let name="str2" type="string"></let>`<br><br>`<function name="string. left" return="str2"> str1, 12 </function>`<br>**Result:**<br>str2="A brown bear" |

| Function name | Description |
|---|---|
| **String right** | The function extracts the last nCount character from string 1 and copies this to the return variable.<br><br>**Parameter:**<br>**str1** - string<br>**nCount** - number of characters<br><br>**Syntax:**<br>`<function name="string.right" return="<result string>"> str1, nCount </function>`<br><br>**Example:**<br>`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`<br>`<let name="str2" type="string"></let`<br><br>`<function name="string. right " return="str2"> str1, 10 </function>`<br>**Result:**<br>str2="brown dog." |
| **String middle** | The function extracts the specified number of characters from string 1, starting from the iFirst index, and copies these to the return variable.<br><br>**Parameter:**<br>**str1** - string<br>**iFirst** - start index<br>**nCount** - number of characters<br><br>**Syntax:**<br>`<function name="string.middle" return="<result string>"> str1, iFirst, nCount </function>`<br><br>**Example:**<br>`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`<br>`<let name="str2" type="string"></let`<br><br>`<function name="string. middle " return="str2"> str1, 2, 5 </function>`<br>**Result:**<br>str2="brown" |

| Function name | Description |
|---|---|
| **String length** | The function gives the number of characters in a string.<br><br>**Parameter:**<br>**str1** - string<br><br>**Syntax:**<br>`<function name="string.length" return="<int var>"> str1 </function>`<br><br>**Example:**<br>`<let name="length">0</let>`<br><br>`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`<br>`<function name="string.length" return="length"> str1 </function>`<br>**Result:**<br>length = 31 |
| **Strings to replace** | The function replaces all the substrings found with the new string.<br><br>**Parameter:**<br>**string** - string variable<br>**find string** - string to be replaced<br>**new string** - new string<br><br>**Syntax:**<br>`<function name="<string.replace>"> string, find string, new string </function>`<br><br>**Example:**<br>`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`<br><br>`<function name="string.replace" > str1, _T"a brown dog" , _T"a big salmon"</function>`<br>**Result:**<br>str1 = "A brown bear hunts a big salmon!" |

| Function name | Description |
|---|---|
| **Strings to remove** | The function removes all the substrings found.<br><br>**Parameter:**<br>**string** - string variable<br>**remove string** - substring to be deleted<br><br>**Syntax:**<br>`<function name="string.remove"> string, remove string </function>`<br><br>**Example:**<br>`<let name="index">0</let>`<br><br>`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`<br>`<function name="string.remove" > str1, _T"a brown dog" </function`<br>**Result:**<br>str1 = "A brown bear hunts" |
| **Strings to insert** | The function inserts a string at the index specified.<br><br>**Parameter:**<br>**string** - string variable<br>**index** - index (zero based)<br>**insert string** - string to be inserted<br><br>**Syntax:**<br>`<function name="string.insert"> string, index, insert string </function>`<br><br>**Example:**<br>`<let name="str1" type="string">A brown bear hunts. </let>`<br>`<let name="str2" type="string">a brown dog</let>`<br><br>`<function name="string.insert" > str1, 19, str2</ function>`<br>**Result:**<br>str1 = "A brown bear hunts a brown dog" |

| Function name | Description |
|---|---|
| **String delete** | The function deletes the defined number of characters starting from the start position specified.<br><br>**Parameter:**<br>**string** - string variable<br>**start index** - start index (zero based)<br>**nCount** - number of characters to be deleted<br><br>**Syntax:**<br>`<function name="string.delete"> string, start index , nCount </function>`<br><br>**Example:**<br>`<let name="str1" type="string">A brown bear hunts. </let>`<br><br>`<function name="string.delete" > str1, 2, 5</ function>`<br>**Result:**<br>str1 = "A bear hunts" |
| **String find** | The function searches the transferred string for the first match with the substring.<br>If the substring is found, the function provides the index to the first character (starting with zero) or, failing this, -1.<br><br>**Parameter:**<br>**string** - string variable<br>**find string** - string to be found<br>**startindex** – start index (optional)<br><br>**Syntax:**<br>`<function name="string.find" return="<int val>"> str1, find string </function>`<br><br>**Example:**<br>`<let name="index">0</let>`<br>`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`<br>`<function name="string.find" return="index"> str1, _T"brown" </function>`<br>**Result:**<br>Index = 2<br>**or**<br>`<function name="string.find" return="index"> str1, _T"brown", 1 </function>` |

| Function name | Description |
|---|---|
| **String reverse find** | The function searches the transferred string for the last match with the substring.<br>If the substring is found, the function provides the index to the first character (starting with zero) or, failing this, -1.<br><br>**Parameter:**<br>**string** - string variable<br>**find string** - string to be found<br>**startindex** – start index (optional)<br><br>**Syntax:**<br>`<function name="string.reversefind" return="<int val>"> str1, find string </function>`<br><br>**Example:**<br>`<let name="index">0</let>`<br>`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`<br>`<function name="string.reversefind" return="index"> str1, _T"brown" </function>`<br>**Result:**<br>Index = 21<br>**or**<br>`<function name="string.reversefind" return="index"> str1, _T"brown", 10 </function>`<br>**Result:**<br>Index = 2 |
| **String trim left** | The function trims the starting characters from a string.<br><br>**Parameter:**<br>**str1** - string variable<br><br>**Syntax:**<br>`<function name="string.trimleft" > str1 </function>`<br><br>**Example:**<br>`<let name="str1" type="string">  test trim left</let>`<br>`<function name="string.trimleft" > str1 </function>`<br>**Result:**<br>str1 = "test trim left" |

| Function name | Description |
|---|---|
| **String trim right** | The function trims the closing characters from a string.<br><br>**Parameter:**<br>**str1** - string variable<br><br>**Syntax:**<br>`<function name="string.trimright" > str1 </ function>`<br><br>**Example:**<br>`<let name="str1" type="string"> test trim right </let>`<br>`<function name="string.trimright" > str1 </ function>`<br>**Result:**<br>str1 = "test trim right" |
| **Sine** | The function calculates the sine of the value transferred in degrees.<br><br>**Parameter:**<br>**double** - angle<br><br>**Syntax:**<br>`<function name="sin" return="<double val>"> double </function>`<br><br>**Example:**<br>`<let name= "sin_val" type="double"></let>`<br>`<function name="sin" return="sin_val"> 20.0 </ function>` |
| **Cosine** | The function calculates the cosine of the value transferred in degrees.<br><br>**Parameter:**<br>**double** - angle<br><br>**Syntax:**<br>`<function name="cos" return="<double val>"> double </function>`<br><br>**Example:**<br>`<let name= "cos_val" type="double"></let>`<br>`<function name="cos" return="cos_val"> 20.0 </ function>` |

| Function name | Description |
|---|---|
| **Tangent** | The function calculates the tangent of the value transferred in degrees.<br><br>**Parameter:**<br>**double** - angle<br><br>**Syntax:**<br>`<function name="tan" return="<double val>"> double </function>`<br><br>**Example:**<br>`<let name= "tan_val" type="double"></let>`<br>`<function name="tan" return="tan_val"> 20.0 </ function>` |
| **ARCSIN** | The function calculates the arcsine of the value transferred in degrees.<br><br>**Parameter:**<br>**double** - x in the range from -PI/2 to +PI/2<br><br>**Syntax:**<br>`<function name="arcsin" return="<double val>"> double </function>`<br><br>**Example:**<br>`<let name= "arcsin_val" type="double"></let>`<br>`<function name="arcsin" return=" arcsin_val"> 20.0 </function>` |
| **ARCOS** | The function calculates the arccosine of the value transferred in degrees.<br><br>**Parameter:**<br>**double -** x in the range from -PI/2 to +PI/2<br><br>**Syntax:**<br>`<function name="arcos" return="<double val>"> double </function>`<br><br>**Example:**<br>`<let name= "arccos_val" type="double"></let>`<br>`<function name="arccos" return=" arccos_val"> 20.0 </function>` |

| Function name | Description |
|---|---|
| **ARCTAN** | The function calculates the arctan of the value transferred in degrees.<br><br>**Parameter:**<br>**double** - arctan of y/x<br><br>**Syntax:**<br>`<function name="arctan" return="<double val>"> double </function>`<br><br>**Example:**<br>`<let name= "arctan_val" type="double"></let>`<br>`<function name="arctan" return="arctan_val"> 20.0 </function>` |
| **DLL load** | The function loads an additional user DLL to the memory.<br><br>**Parameter:**<br>**dll_name** - DLL name<br>**class_name** - name of the function class<br><br>**Syntax:**<br>`<function name="dll.load"> dll_name, class_name </function>`<br><br>**Example:**<br>`<function name="dll.load"> _T"customer.dll", _T"customer" </function>` |
| **DLL function** | The function calls a function from a user DLL. All parameters listed after the parameter ID are transferred to the function called.<br><br>**Parameter:**<br>**class_name** - name of the function class<br>**id** - of the function<br>**parameter** - maximum seven function parameters (string variables)<br><br>**Syntax:**<br>`<function name="dll.function"> class_name, id, parameter1, parameter2</function>`<br><br>**Example**<br>`<function name="dll.function"> _T"customer", 290, _T"par1", _T"par2"</function>` |
| File processing | |

| Function name | Description |
|---|---|
| **Loading a file** | The function loads the contents of the file specified to a string variable.<br><br>**Attribute:**<br>**Return** - name of the local variable<br><br>**Parameter:**<br>**Progname** - file name<br><br>**Syntax:**<br>`<function name="doc.readfromfile" return="<string var>"> progname </function>`<br><br>**Example:**<br>`<let name = "my_var" type="string" ></let>`<br><br>`<function name=" doc.readfromfile " return="my_var"> _T"\spf\test.mpf" </function>` |
| **Writing to a file** | The function writes the contents of a string variable to the file specified.<br><br>**Parameter:**<br>**progname** - file name<br>**str1** - string<br><br>**Syntax:**<br>`<function name="doc.writetofile" > progname, str1 </function>`<br><br>**Example:**<br>`<let name = "my_var" type="string" > file content </let>`<br><br>`<function name="doc.writetofile">_T"\spf \test.mpf",  my_var  </function>` |
| **Deleting a file** | The function removes the file specified from the directory.<br><br>**Parameter:**<br>**progname** - file name<br><br>**Syntax:**<br>`<function name="doc.remove" > progname </function>`<br><br>**Example:**<br>`<function name="doc.remove">_T"\mpf\test.mpf" </function>` |

| Function name | Description |
|---|---|
| **Exist** | If the file exists, the function returns the value 1.<br><br>**Parameter:**<br>**progname** - file name<br><br>**Syntax:**<br>`<function name="doc.exist" return="<int_var>" > progname </function>`<br><br>**Example:**<br>`<let name ="exist">0</let>`<br><br>`<function name="doc.exist" return="exist">_T"\mpf \test.mpf" </function>` |
| **NC program selection** | The function selects the program specified for execution. The program must be stored in the NC file system.<br>**Parameter:**<br>**progname** - file name<br><br>**Syntax:**<br>`<function name="ncfunc.select"> progname </ function>`<br><br>**Example:**<br>`<function name="ncfunc.select"> _T"\mpf\test.mpf" </function>` |

| Function name | Description |
|---|---|
| **Setting an individual bit** | The function is used to manipulate individual bits of the specified variables.<br>The bits can either be set or reset.<br><br>**Syntax:**<br>`<function name="ncfunc.bitset" refvar="address" value="set/reset" > bit0, bit1, … bit9 </function>`<br><br>**Attributes:**<br>**refvar** - specifies the name of the variable, in which the bit combination should be written<br>**value** – bit value, value range 0 and 1<br><br>**Values:**<br>The bit numbers starting with zero should be transferred as function values.<br>A maximum of 10 bits per call can be modified.<br><br>**Example:**<br>`<function name="ncfunc.bitset" refvar="nck/Channel/Parameter/R[1]" value="1" > 0, 2, 3, 7 </function>`<br><br>`<function name="ncfunc.bitset" refvar="nck/Channel/Parameter/R[1]" value="0" > 1, 4 </function>` |
| **Delete control** | The function deletes the specified picture control.<br><br>**Syntax:**<br><function name="control.delete"> control name </function><br>or<br><function name="ncfunc.bitset" refvar="$MN_USER_DATA_HEX" value="1" > 0, 2, 3, 7 </function><br><br>**Attribute:**<br>**name** – function name<br><br>**Value:**<br>**control name** – name of the control<br><br>**Example:**<br>`<function name="<control.delete>"> _T"my_editfield" </function>` |

| Function name | Description |
|---|---|
| **Add Item** | The function inserts a new element at the end of the list.<br>Note:<br>The function is only available for the control types "listbox" and "graphicbox".<br><br>**Syntax:**<br>`<function name="control.additem"> control name, item </function>`<br><br>**Attribute:**<br>**name** – function name<br><br>**Values:**<br>**control name** – control name<br>**item** - expression to be inserted<br>**itemdata** - integer value; defined by the user<br><br>**Example:**<br>`<let name ="itemdata">1</let>`<br>…<br>…<br>…<br>`<op> item_string = _T"text1" </op>`<br>`<function name="control.additem">_T"listbox1", item_string, itemdata`<br>`</function>` |

| Function name | Description |
|---|---|
| **Insert Item** | The function inserts a new element at the specified position.<br>Note:<br>The function is only available for the control types "listbox" and "graphicbox".<br><br>**Syntax:**<br>`<function name="control.insertitem"> control name, index, item, itemdata </function>`<br><br>**Attribute:**<br>**name** – function name<br><br>**Values:**<br>**control name** – control name<br>**index** – position starting with zero<br>**item** - expression to be inserted<br>**itemdata** - integer value; defined by the user<br><br>**Example:**<br>`<let name ="itemdata">1</let>`<br>…<br>…<br>…<br>`<op> item_string = _T"text2" </op>`<br>`<function name="control.insertitem">_T"listbox1", 1, item_string, itemdata </function>` |
| **Delete item** | The function deletes an element at the specified position.<br>Note:<br>The function is only available for the control types "listbox" and "graphicbox".<br><br>**Syntax:**<br>`<function name="control.deleteitem"> control name, index </function>`<br><br>**Attribute:**<br>**name** – function name<br><br>**Values:**<br>**control name** – control name<br>**index**– index starting at 0<br><br>**Example:**<br>`<function name="control.deleteitem">_T"listbox1", 1</function>` |

| Function name | Description |
|---|---|
| **Load Item** | The function inserts a list of expressions into the control.<br>The function is only available for the control types "listbox" and "graphicbox".<br><br>**Syntax:**<br>`<function name="control.loaditem"> control name, list </function>`<br><br>**Attribute:**<br>**name** – function name<br><br>**Values:**<br>**control name** – control name<br>**list**- string variable<br><br>**Structure of the list:**<br>The list contains a number of expressions, which must be separated from one another using a \n.<br><br>**Example:**<br>`<let name="item_string" type="string"></let>`<br>`<let name="plotlist" type="string"></let>`<br>…<br>…<br>`<print name ="item_string" text="p; %f; %f; %f; %f \n">s_z, s_x</print>`<br>`<op>plotlist = plotlist + item_string</op>`<br>`<print name ="item_string" text="l; %f; %f; %f; %f\n">s_z, s_x, e_z, e_x </print>`<br>`<op>plotlist = plotlist + item_string</op>`<br>`<op> s_x = e_x </op>`<br>`<op> s_z = e_z</op>`<br>`<op> e_x = s_x + 10 </op>`<br>`<op> e_z = s_z - 100 </op>`<br>`<print name ="item_string" text="l; %f; %f; %f; %f\n">s_z, s_x, e_z, e_x </print>`<br>`<op>plotlist = plotlist + item_string</op>`<br><br>`<function name="control.loaditem">_T"gbox", plotlist</function>` |

| Function name | Description |
|---|---|
| **Empty** | The function deletes the contents of the specified list box or graphic box controls.<br><br>**Syntax:**<br>`<function name="control.empty"> control name, </function>`<br><br>**Attribute:**<br>**name** – function name<br><br>**Values:**<br>**control name** – Control name<br><br>**Example:**<br>`<function name=" control.empty">_T"listbox1"</function>` |
| **Get focus** | The function supplies the name of the control, which has the input focus.<br><br>**Syntax:**<br>`<function name="control.getfocus" return="focus_name" />`<br><br>**Attributes:**<br>**name** – function name<br>**return** – a string variable should be specified, into which the control name is copied.<br><br>**Example:**<br>`<let name>="focus_field" type="string"></let>`<br>`<function name="control.getfocus" return="focus_field"/>` |

| Function name | Description |
|---|---|
| **Set focus** | The function sets the input focus to the specified control.<br>The Controlname should be transferred as text expression of the function.<br><br>**Syntax:**<br>`<function name="control.setfocus"> control name </function>`<br><br>**Attribute:**<br>**name** – function name<br><br>**Value:**<br>**control name** – name of the control<br><br>**Example:**<br>`<function name="control.setfocus" ">_T"listbox1"</function>` |
| **Get cursor selection** | For a list box, the function supplies the cursor index.<br>The Controlname should be transferred as text expression of the function.<br><br>**Syntax:**<br>`<function name="control.getcurssel" retvar="var"> control name </function>`<br><br>**Example:**<br>`<let name>="index"></let>`<br>`<function name="control.getcurssel" ">_T"listbox1"</function>` |
| **Set cursor selection** | For a list box, the function sets the cursor to the appropriate line.<br>The Controlname should be transferred as text expression of the function.<br><br>**Syntax:**<br>`<function name="control.setcurssel" > control name, index</function>`<br><br>**Example:**<br>`<let name>="index">2</let>`<br>`<function name="control.setcurssel" ">_T"listbox1",index</function>` |

| Function name | Description |
|---|---|
| **Get Item** | For a list box, the function copies the contents of the selected line to the specified variable.<br>A string variable should be specified as reference variable.<br>The Controlname should be transferred as text expression of the function.<br><br>**Syntax:**<br>`<function name="control.getitem" return="var">`<br>`control name, index </function>`<br><br>**Example:**<br>`<let name>="index">2</let>`<br>`<let name>="item" type="string"></let>`<br><br>`<function name="control.getitem" return="item"`<br>`">_T"listbox1",index</function>` |
| **Get Item Data** | For a list box, the function copies the user-specific allocated value of an element to the specified variable.<br>For an edit control, the function copies the user-specific allocated value (item_data) to the specified variable.<br>An integer variable should be specified as reference variable.<br>The Controlname should be transferred as text expression of the function.<br><br>**Syntax:**<br>`<function name="control.getitemdata" return="var">`<br>`control name, index </function>`<br><br>**Example:**<br>`<let name>="index">2</let>`<br>`<let name>="itemdata"></let>`<br><br>`<function name="control.getitemdata"`<br>`return="itemdata" ">_T"listbox1",index</function>` |
| **Abs** | This function returns the absolute value of the specified number.<br><br>**Syntax:**<br>`<function name="abs" return="var"> value </`<br>`function>` |
| **SDEG** | The function converts the specified value into degrees.<br><br>**Syntax:**<br>`<function name="sdeg" return="var"> value </`<br>`function>` |
| **SRAD** | The function converts the specified value into RADian.<br><br>**Syntax:**<br>`<function name="srad" return="var"> value </`<br>`function>` |

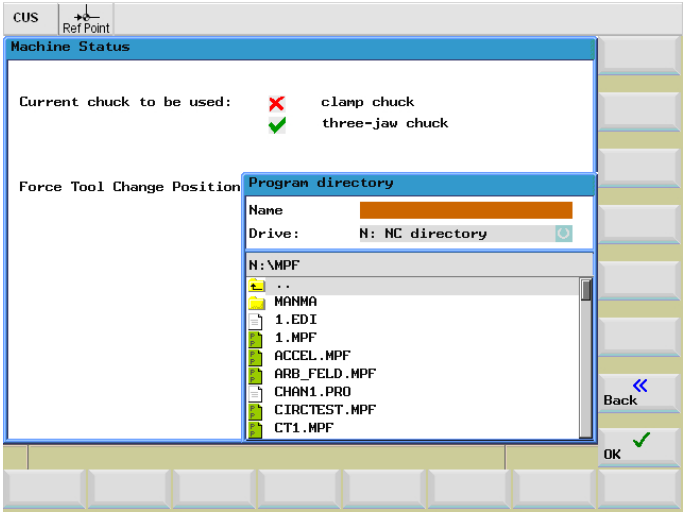
| Function name | Description |
|---|---|
| **SQRT** | The function calculates the square root of the specified value.<br><br>**Syntax:**<br>`<function name="sqrt" return="var"> value </function>` |
| **ROUND** | The function rounds of the transferred number to the specified number of decimal places. If the number of decimal places is not specified, then the function rounds off the number, taking into account the first decimal place.<br><br>**Syntax:**<br>`<function name="round" return="var"> value, nDecimalPlaces </function>` |
| **FLOOR** | The function supplies the largest possible integer value, which is less than or equal to the transferred value.<br><br>**Syntax:**<br>`<function name="floor" return="var"> value </function>` |
| **CEIL** | The function supplies the smallest possible integer value, which is greater than or equal to the transferred value.<br><br>**Syntax:**<br>`<function name="ceil" return="var"> value </function>` |
| **LOG** | The function calculates the logarithm of the specified value.<br><br>**Syntax:**<br>`<function name="log" return="var"> value </function>` |
| **LOG10** | The function calculates the common (decadic) logarithm of the specified value.<br><br>**Syntax:**<br>`<function name="log10" return="var"> value </function>` |
| **POW** | The function calculates the value "$a^b$".<br><br>**Syntax:**<br>`<function name="pow" return="var"> a, b </function>` |
| **MIN** | The function compares the transferred value and returns the lower of the values.<br><br>**Syntax:**<br>`<function name="min" return="var"> value1, value2 </function>` |

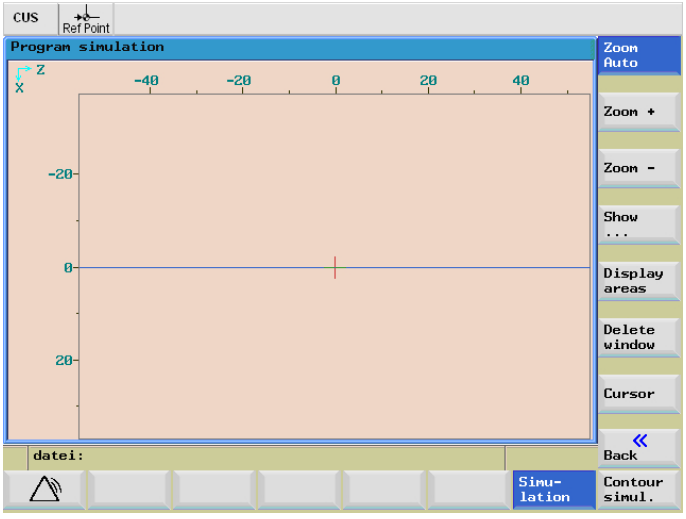
| Function name | Description |
|---|---|
| **MAX** | The function compares the transferred value and returns the higher of the values.<br><br>**Syntax:**<br>`<function name="max" return="var"> value1, value2 </function>` |
| **RANDOM** | The function returns a pseudo random number.<br><br>**Syntax:**<br>`<function name="random" return="var" </function>` |
| **MMC** | **Function**<br>User-defined dialog windows (dialog screens) from the part program can be displayed with the MMC command. The appearance of the dialog window is specified through purely textual configuration (XML file in the F:\appl directory (manufacturer drive)); here, the HMI software remains unchanged.<br>**Syntax**<br>MMC ("operating area, command, XML script file, menu name, reserved, reserved, display time or acknowledgment variable, reserved", "acknowledgment mode")<br>**Description**<br>• MMC<br>Calling the dialog window interactively from the part program on the HMI.<br>• EasyXml<br>Identification for user dialogs that are to be started from a part program.<br>• XML_ON or XML_OFF<br>Command: Display selection or display deselection<br>• XML file<br>Name of the XML script file<br>• Menu<br>Name of the menu tag that manages the dialog to be displayed<br>• reserved<br>reserved for HMI Operate<br>• reserved<br>reserved for HMI Operate<br>• Time<br>Display time of the dialog for acknowledgments mode "N"<br>• reserved<br>reserved for HMI Operate<br>• Acknowledgement mode<br>"S" synchronous, acknowledgement via the "OK" softkey<br>"N" asynchronous, dialog closes after the agreed time |

| Function name | Description |
|---|---|
| **MMC** continued | **Example of synchronous call:**<br>NC instruction<br>`MMC("EASYXML oder CYCLES", XML_ON, mmc_cmd.xml, cmd1,,,,,","S")`<br>`Datei: mmc_cmd.xml`<br><br>`<menu name = "cmd1">`<br>`<open_form name = "cmd1_form" />`<br>`<softkey_ok>`<br>`<close_form />`<br>`</softkey_ok>`<br>`</menu>`<br><br>`<form name = "cmd1_form" xpos ="12" ypos="100" width="500" height="240">`<br>`<init>`<br>`</init>`<br>`<paint>`<br>`</paint>`<br>`</form>`<br><br>**Example of asynchronous call (acknowledgment not expected):**<br>NC instruction<br>`MMC("EASYXML oder CYCLES", PICTURE_ON, mmc_cmd.xml, cmd1,,,10,,","N")`<br>`Datei: mmc_cmd.xml <menu name = "cmd1">`<br><br>`<open_form name = "cmd1_form" />  <softkey_ok>`<br>`<close_form />  </softkey_ok>`<br>`</menu>`<br><br>`<form name = "cmd1_form" xpos ="12" ypos="100" width="500" height="240">`<br>`<init>`<br>`</init>`<br>`<paint>`<br>`</paint>`<br>`</form>` |

| Function name | Description |
|---|---|
| **String find** | The function searches the transferred string for the first match with the substring.<br>If the substring is found, the function provides the index to the first character (starting with zero) or, failing this, -1.<br><br>**Parameter:**<br>**string** - string variable<br>**findstring** - string to be found<br>**start index** – start index (optional)<br><br>**Syntax:**<br>`<function name="string.find" return="<int val>"> str1, find string </function>`<br><br>**Example:**<br>`<let name="index">0</let> <let name="str1" type="string">A brown bear hunts a brown dog. </let> <function name="string.find" return="index"> str1, _T"brown" </function>`<br><br>**Result:**<br>Index = 2<br><br>**or**<br>`<function name="string.find" return="index"> str1, _T"brown", 1 </function>` |

| Function name | Description |
|---|---|
| **String reverse find** | The function searches the transferred string for the last match with the substring.<br>If the substring is found, the function provides the index to the first character (starting with zero) or, failing this, -1.<br><br>**Parameter:**<br>**string** - string variable<br>**find string** - string to be found<br>**start index** – start index (optional)<br><br>**Syntax:**<br>`<function name="string.reversefind" return="<int val>"> str1, find string </function>`<br><br>**Example:**<br>`<let name="index">0</let>`<br>`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`<br>`<function name="string.reversefind" return="index"> str1, _T"brown" </function>`<br><br>**Result:**<br>Index = 21<br><br>**or**<br>`<function name="string.reversefind" return="index"> str1, _T"brown" 10 </function>`<br><br>**Result:**<br>Index = 2 |
| **Dialogs** | |

| Function name | Description |
|---|---|
| **Program selection dialog** | The function opens a dialog for selecting a part program.<br>The selected program name is copied to the reference variable specified.<br>The dialog can only be called from within a softkey tag.<br><br><br>**Syntax:**<br>`<function name= "doc.fileselect" return="string var" />`<br>**Example:**<br>`<let name = "program_content" type="string" > </let>`<br>`<let name = "curr_program" type="string" > </let>`<br>`<function name= "doc.fileselect" return="curr_program" />`<br>`<function name="doc.readfromfile " return=" program_content" > curr_program </function>` |

| Function name | Description |
|---|---|
| **Program simulation dialog** | The function opens program simulation in the form of a pop-up dialog.<br>If a program name is specified, then the program is automatically selected. Otherwise, the simulation represents the traversing motion of the currently selected program.<br>The dialog can only be called from within a softkey tag.<br>The "Back" softkey function returns the user to the previous dialog.<br><br>**Syntax:**<br>`<function name= "ncfunc.progsimulation"> prog name </function>`<br>**Example:**<br>`<let name = "curr_program" type="string" > </let>`<br>`<function name= "doc.fileselect" return="curr_program" />`<br>`<function name= "ncfunc.progsimulation"> curr_program </function>` |

# Generating commissioning dialogs

# 2

## 2.1 Overview of functions

### Purpose

The "Startup Wizard" allows additional devices to be simply commissioned, activated, deactivated and tested. The available equipment and device states are displayed in a list by the control system. The system can manage a maximum of 64 devices.

Softkeys are used to activate or deactivate a device.

The "Startup Wizard" function is available in the <SYSTEM> operating area.

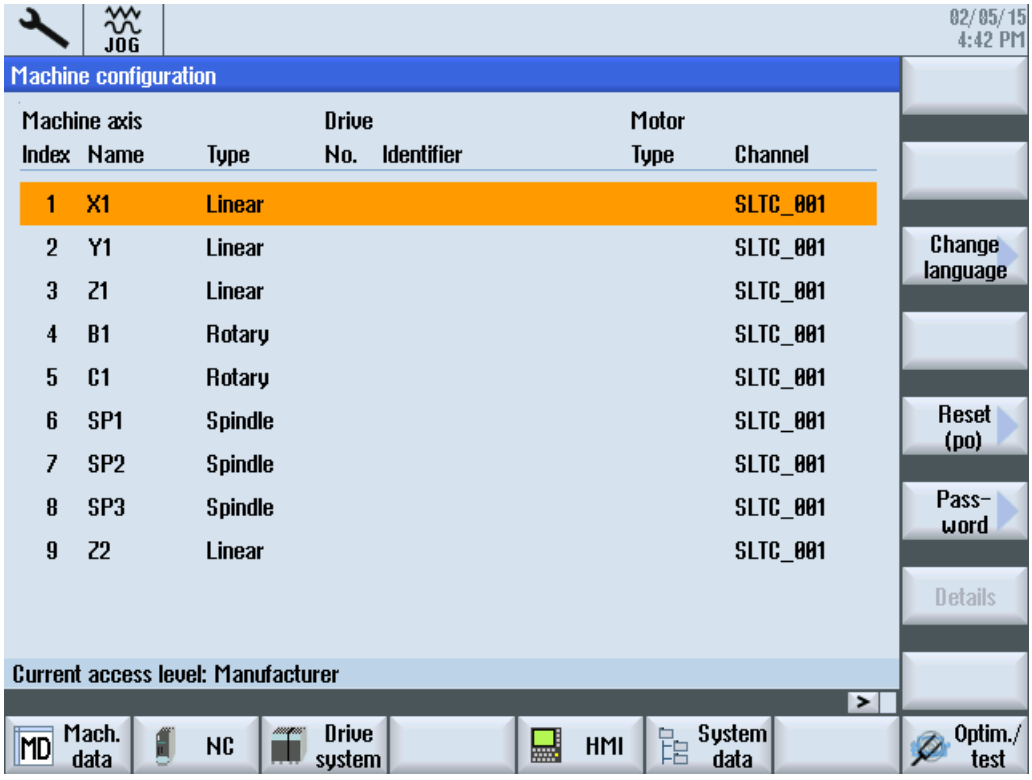



Figure 2-1 The SYSTEM main screen with active "Startup Wizard" softkey

## Configuration

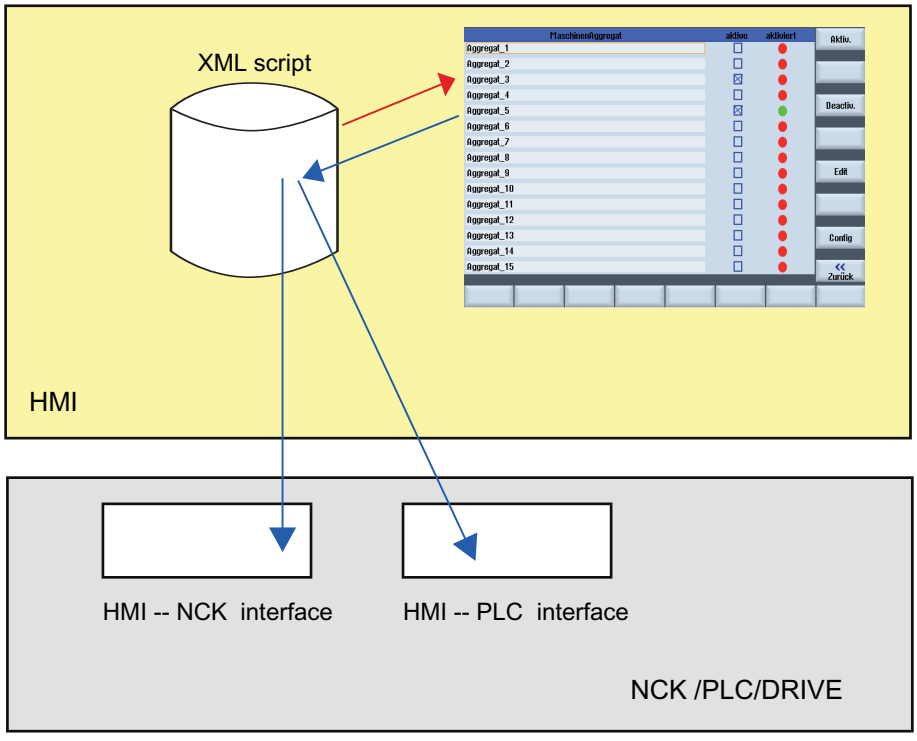



Figure 2-2 Mode of operation of the "Startup Wizard"

To use the "Startup Wizard", the following functions should be configured by the machine manufacturer:

- **PLC ↔ HMI interface**
  The optional devices are managed via the interface between the user interface and the PLC.
- **Script processing**
  The machine manufacturer saves the sequences to be executed for commissioning, activating, deactivating or testing a device, in a statement script.
- **Parameter dialog (optional)**
  The parameter dialog shows device information that is saved in the script file.

## Storage of the files

The "Startup Wizard" files are stored on the system CompactFlash card in the "dvm" (machine builder) directory.

| File | Name | Target directory |
|---|---|---|
| Text file | oem_aggregate_xxx.ts | `card\oem\sinumerik\hmi\lmg` |
| Script file | agm.xml | `card\oem\sinumerik\hmi\dvm` |

| File | Name | Target directory |
|---|---|---|
| Archive file | Any | `card\oem\sinumerik\hmi\dvm \archives` |
| PLC user program | Any | PLC |

# 2.2 Configuration in the PLC user program

## Loading configurations

The configurations created are transferred to the manufacturer directory of the control, with the script and text file. Additionally, the corresponding PLC user program should be loaded.

## Programming the equipment

Communication between the operator component and the PLC takes place in the PLC user program via data block VB99050000, in which 128 words are reserved for the device management.

PLC words are assigned beginning with Device 1:

| Data block | Device designation |
|---|---|
| DB9905.DBB0 | Device 1 |
| DB9905.DBB4 | Device 2 |
| ... | ... |
| DB9905.DBB192 | Device 49 |
| DB9905.DBB196 | Device 50 |

Four bytes with the following meanings are used for each device:

| Byte | Bit | Description | |
|---|---|---|---|
| 0 | 0 | == 1 | Device has been started up (HMI acknowledgment) |
| | 1 | == 1 | Device is to be activated (HMI request) |
| | 2 | == 1 | Device is to be deactivated (HMI request) |
| | 3-7 | Reserved | |
| 1 | 0-7 | Reserved | |
| 2 | 0 | == 1 | Device is active (PLC acknowledgment) |
| | 1 | == 1 | Device has an error |
| | 2-7 | Reserved | |
| 3 | 0-7 | Unique identifier for the device | |

## Storage of the files

The Easy Extend files are stored on the system CompactFlash card in the "oem" (MANUFACTURER) and "oem_i" (INDIVIDUAL) directory.

| File | Name | Target directory |
|---|---|---|
| Text file | oem_aggregate_xxx.ts | /oem/sinumerik/hmi/lng/<br>/oem_i/sinumerik/hmi/lng/ |
| Script file | agm.xml | /oem/sinumerik/hmi/dvm<br>/oem_i/sinumerik/hmi/dvm |
| Archive file | Any | /oem/sinumerik/hmi/dvm/archives<br>/oem_i/sinumerik/hmi/dvm/archives |
| PLC user program | Any | PLC |

## General sequence

The machine manufacturer must execute the following steps to make the required data available:

1. Creating a PLC user program which activates the device during activation on the PLC.
2. Commissioning of the "standard machine" followed by backup of the data in a series startup archive.
3. Installation of the devices, commissioning, followed by read-out of the data as a differential series startup archive.

**Note**

**Changing the machine configuration**

Should there be any need to edit the drive machine data, this should be adapted in the control first. This procedure should be repeated for all devices and constellations.

## Adding axes

If the machine is extended with machine axes, it is important to install the drive objects (DO) in a fixed sequence because the series startup archive contains the constellation of the machine manufacturer's reference machine and cannot be applied if the sequence is changed.

It is recommended that the following settings be selected for the "control components":

- NC data
- PLC data
- Drive data
  - ACX format (binary)

# 2.3 Display on the user interface

## Dialogs on the user interface

The following dialogs are available for "Startup Wizard":

- The control offers a **configurable dialog**, in which the available devices are shown.
- If first commissioning has not taken place yet, the control opens the **commissioning dialog**.

If a commissioning procedure (XML instruction: "START_UP") has been programmed, and the device has still not been commissioned, then the control starts the commissioning procedure.

This involves a complete data backup before the series startup archives saved in the script file are read in.

In the event of an error, the commissioning engineer can decide whether to roll back the commissioning procedure or to rectify possible errors in machine configurations manually.

- Commissioning can be aborted early with the "Cancel" function. The control then copies the previously saved commissioning files back.

If the machine has to be switched off after successful completion of the commissioning, the XML statement "POWER_OFF" can be used to program that a corresponding message is output on the control.

# 2.4 Creating language-dependent texts

## Structure of text file

The XML files with the language-dependent texts must be created in UTF8 format:

**Example oem_aggregate_eng.ts**

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TS>
<TS>
       <context>
       <name>EASY_EXTEND</name>
           <message>
           <source>DEVICE_ONE</source>
           <translation>Device one</translation>
           </message>
           <message>
           <source>DEVICE_TWO</source>
           <translation>Device two</translation>
           </message>
       </context>
</TS>
```

**Example oem_aggregate_deu.ts**

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TS>
<TS>
       <context>
       <name>EASY_EXTEND</name>
            <message>
           <source>DEVICE_ONE</source>
           <translation>Device one</translation>
           </message>
           <source>DEVICE_TWO</source>
           <translation>Device two</translation>
           </message>
       </context>
</TS>
```

## 2.5 User example for a power unit

### Activating the drive object

The drive object to be activated has already been commissioned and deactivated again by the machine manufacturer, to market the axis (axes) as an option.

To activate the axis carry out the following steps:

- Activate the drive object via p0105.
- 2. Enable the axis in the channel machine data.
- Back up the drive machine data via p0971.
- Wait until the data has been written.
- Restart the NCK and the drives.

**Programming:**

```
<DEVICE>
  <list_id>1</list_id>
  <name> "Activate the drive" </name>

  <SET_ACTIVE>
    <data name = "drive/dc/p105[DO5]">1</data>
    <data name = "$MC_AXCONF_MACHAX_USED[4]">5</data>
    <data name = "drive/dc/p971[DO5]">1</data>
    <while>
      <condition> "drive/dc/p971[DO5]" !=0 </condition>
    </while>
    <control_reset resetnc ="true" resetdrive = "true"/>
  </SET_ACTIVE>

  <SET_INACTIVE>
    <data name = "drive/dc/p105[DO5]">0</data>
    <data name = "$MC_AXCONF_MACHAX_USED[4]">0</data>
    <data name = "drive/dc/p971[DO5]">1</data>
    <while>
      <condition> "drive/dc/p971[DO5]" !=0 </condition>
    </while>
    <control_reset resetnc ="true" resetdrive = "true"/>
  </SET_INACTIVE>

</DEVICE>
```

### Activating the PLC-controlled device

The device is addressed via output byte 10 and signals data set ready to the PLC via input byte 9.

The output byte is set to the specified coding for activation. The WHILE loop then waits for the data set ready of the device.

**Programming:**

```
<SET_ACTIVE>
  <DATA name = "plc/qb10"> 8 </DATA>
  <while>
    <condition> "plc/ib9" !=1 </condition>
  </while>
</SET_ACTIVE>
```

## 2.6 Script language

**Note**

All of the script elements described in the Generating user dialogs (Page 5) function form the basis for the "Startup Wizard" function. Additional script elements are defined to manage additional devices.

### Program parts of the script

The script is divided into the following areas:

- "Startup Wizard" - frame
- Frame to define the actions that can be executed for a device
- Identifier for the device
- Identifier for commissioning the device
- Identifier for activating the device
- Identifier for deactivating the device
- Identifier for testing the device
- Identifier for the parameter dialog

The individual tags are described in the following chapters.

## Description

| Identifier <tag> | Meaning |
|---|---|
| AGM | Identifier for the "Startup Wizard" |
| DEVICE | Identifier for the description of the device.<br><br>**Attributes:**<br>• **option_bit**<br>The device is assigned a fixed bit number for the option management. |
| NAME | The identifier specifies the name of the device to be displayed in the dialog.<br>If a text reference is used, the dialog displays the text which is saved for the identifier. |
| START_UP | The identifier contains a description of the sequences required for commissioning the device. |
| SET_ACTIVE | The identifier contains a description of the sequences required to activate the device.<br>**Attributes:**<br>• **timeout**<br>The attribute permits a timeout to be specified in seconds. The system interrupts processing if the script has still not been completed after this time. |
| SET_INACTIVE | The identifier contains a description of the sequences required to shut down the device.<br>**Attributes:**<br>• **timeout**<br>The attribute permits a timeout to be specified in seconds. The system interrupts processing if the script has still not been completed after this time. |
| TEST | The identifier contains the statements for testing the operating capability of a device.<br>**Attributes:**<br>• **timeout**<br>The attribute permits a timeout to be specified in seconds. The system interrupts processing if the script has still not been completed after this time. |
| UID | Unique numerical identifier to identify the device in the PLC ↔ HMI interface. |
| VERSION | Identifier for a version |

## Negative acknowledgment of the function execution

With the automatically provided variable "$actionresult", the system can inform the XML parser of a negative execution result. If the value is set to zero, the parser aborts the function processing.

## Example

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE AGM>
<AGM>                                   Identifier for the "Startup Wizard"
<DEVICE>
  <NAME> Device 1 </NAME>               Identifier for the device
  <START_UP>                            Identifier for commissioning the device
  …
  </START_UP>
  <SET_ACTIVE>                          Identifier for activating the device
  …
  </SET_ACTIVE>
  <SET_INACTIVE>                        Identifier for deactivating the device
  …
  </SET_INACTIVE>
  <TEST>                                Identifier for testing the device
  …
  </TEST>
</DEVICE>
…
</AGM>
```

## 2.6.1 CONTROL_RESET

### Description

This identifier allows one or more control components to be restarted. Execution of the script is only continued when the control has resumed cyclic operation.

### Programming

| | | |
|---|---|---|
| Identifier: | **CONTROL_RESET** | |
| Syntax: | <CONTROL_RESET resetnc="TRUE" /> | |
| Attributes: | resetnc="true" | The NC component is restarted. |
| | resetdrive="true" | The drive components are restarted. |

## 2.6.2 FILE

### Description

The identifier enables standard archives to be read in or created.

- Reading in an archive:
  The file name of the archive must be specified for reading in an archive.
- Creating an archive:
  If the attribute create= "true" is specified, the function creates a standard archive (*.arc) under the specified name and stores the file in the …/dvm/archives directory.

### Programming

| | | |
|---|---|---|
| Identifier: | **FILE** | |
| Syntax: | <file name ="<archive name>" /> | |
| | <file name ="<archive name>" create="true" class="<data classes>" group="<area>" /> | |
| Attributes: | name | Identifier for the file name |

| | |
|---|---|
| create | A commissioning archive is created under the specified name in the …/dvm/archives/ directory. |
| group | Specifies the data groups that are to be contained in the archive. If several data groups are to be saved, the groups should be separated by a blank.<br>The following data groups can be contained in the archive:<br>• NC<br>• PLC<br>• HMI<br>• DRIVES |

### Example

```
<!-- Create data class archive -->
<file name="user.arc" create="true"
group="nc plc hmi" />

<!—Read archive into the control->
<file name="user.arc" />
```

## 2.6.3 OPTION_MD

### Description

The identifier allows option machine data to be redefined. As delivered, the system uses MD14510 $MN_USER_DATA_INT[0] to $MN_USER_DATA_INT[3].

If the PLC user program manages the options, the appropriate data words must be provided in a data block or GUD.

The data is structured in bits. Starting with bit 0, there is a fixed assignment of the bits to the listed devices, i.e. bit 0 is assigned to device 1, bit 1 to device 2, etc. If more than 16 devices are managed, the address identifiers of the device groups 1-3 are assigned via the area index.

**Note**

**Converting the value range**

The value range of MD14510 $MN_USER_DATA_INT[i] is from -32768 to +32767. To activate the devices bit-by-bit via the machine data dialog, the bit combination must be converted to decimal representation.

### Programming

| | | |
|---|---|---|
| Identifier: | **OPTION_MD** | |
| Syntax: | Area 0: | |
| | <option_md name = "Address identifier of the data" /> | |
| | OR: | |
| | <option_md name = "Address identifier of the data" index= "0"/> | |
| | Area 1 to 3: | |
| | <option_md name = "Address identifier of the data" index= "Area index"/> | |
| Attributes: | name | Identifier for the address, e.g. $MN_USER_DATA_INT[0] |
| | index | Identifier for the area index:<br>0 (default setting): Device 1 to 16<br>1: Device 17 to 32<br>2: Device 33 to 48<br>3: Device 49 to 64 |

## 2.6.4 PLC_INTERFACE

### Description

This identifier permits the PLC ↔ HMI interface to be redefined. The system expects 128 addressable words.

**Default:** DB9905 (VB99050000)

### Programming

| | | |
|---|---|---|
| Identifier: | **PLC_INTERFACE** | |
| Syntax: | <plc_interface name = "Address identifier of the data" /> | |
| Attributes: | name | Identifier for the address, e.g. "plc/mb170" |

**Example:** `plc/mb170`

## 2.6.5 POWER_OFF

### Description

Identifier for a message prompting the operator to switch the machine off. The message text is permanently saved in the system.

### Programming

| | |
|---|---|
| Identifier: | **POWER_OFF** |
| Syntax: | <power_off /> |
| Attributes: | -- |

## 2.6.6 WAITING

### Description

After a reset of the NC or the drive, there is a wait for the restart of the respective component.

### Programming

| | | |
|---|---|---|
| Identifier: | **WAITING** | |
| Syntax: | <WAITING WAITINGFORNC ="TRUE" /> | |
| Attributes: | waitingfornc="true" | There is a wait for the restart of the NC. |
| | waitingfordrive="true" | There is a wait for the restart of the drive. |

## 2.6.7 XML identifiers for the dialog

### Dialog for the parameterization

A dialog can be configured for each device so that additional parameters can be set or output during runtime. This is displayed by pressing the "Additional parameters" softkey.

All of the script elements described in Chapter Generating user dialogs (Page 5) can be used to generate the dialog.

### Example

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE AGM>
<AGM>
<DEVICE>
  <NAME> Device 1 </NAME>
  <START_UP>
  …
  </START_UP>
  <SET_ACTIVE>
  …
  </SET_ACTIVE>
…
  <FORM>                                  Identifier for a user dialog
    <INIT>
      <CONTROL name = "edit1" .../>       Identifier for an input field
    </INIT>
    <PAINT>                               Identifier for text or image display
      <TEXT>hello world !</TEXT>
    </PAINT>
  </FORM>

</DEVICE>
…
</AGM>
```

## 2.6.8 SOFTKEY_OK, SOFTKEY_CANCEL

### Description

The identifier SOFTKEY_OK overwrites the standard behavior when closing a dialog by means of the "OK" softkey. The identifier SOFTKEY_CANCEL overwrites the standard behavior when closing a dialog by means of the "CANCEL" softkey.

The following functions can be performed within this identifier:

- Data manipulation
- Conditional processing
- Loop processing

## Programming

| | |
|---|---|
| Identifier: | **SOFTKEY_OK** |
| Syntax: | <SOFTKEY_OK><br>…<br></SOFTKEY_OK> |
| Identifier: | **SOFTKEY_CANCEL** |
| Syntax: | <SOFTKEY_CANCEL><br>…<br></SOFTKEY_CANCEL> |

# Index


## C

Configuration file, 7

## G

Generating commissioning dialogs, 111

## M

Menu tree, 7

## S

Start softkey, 7
Startup Wizard, 111

## X

XML
  Operators, 41
  Syntax, 40
XML function No.
  Control form color, 83
XML functions
  Abs, 102
  AddItem, 97
  ARCOS, 92
  ARCSIN, 92
  ARCTAN, 93
  CEIL, 103
  Control local time, 83
  Copy value and read, 78
  Copying and writing a value, 78
  Cosine, 91
  Delete control, 96
  Delete number of characters of a string, 89
  DeleteItem, 98
  Deleting a file, 94
  display resolution, 81
  Empty, 100
  Execute user function, 93
  Exist, 95
  Find substring, 89
  Find substring (reverse), 90
  FLOOR, 103
  Get cursor selection, 101
  getfocus, 100
  GetItem, 102
  GetItemData, 102
  InsertItem, 98
  Load user DLL, 93
  Loading a file, 94
  LoadItem, 99
  LOG, 103
  LOG10, 103
  MAX, 104
  MIN, 103
  MMC, 104
  NC program selection, 95
  Ncfunc bico to int, 81
  Ncfunc int to bico, 81
  Ncfunc is bico str valid, 82
  Ncfunc password, 82
  PI service, 79, 80
  POW, 103
  Program selection dialog, 108
  Program simulation dialog, 109
  RANDOM, 104
  ROUND, 103
  SDEG, 102
  Set cursor selection, 101
  setfocus, 101
  Setting an individual bit, 96
  Sine, 91
  SQRT, 103
  SRAD, 102
  String comparison, 84, 85
  String find, 106
  String reverse find, 107
  string.icmp, 85
  string.insert, 88
  string.left, 85
  string.length, 87
  string.middle, 86
  string.remove, 88
  string.replace, 87
  string.right, 86
  Tangent, 92
  Writing to a file, 94
XML identifier
  AGM, 120
  BOX, 63
  BREAK, 16
  CAPTION, 49

CLOSE, 49
CLOSE_FORM, 49
CONTROL, 50
CONTROL_RESET, 16, 122
CREATE_CYCLE, 70
CREATE_CYCLE_EVENT, 17
DATA, 18
DATA_ACCESS, 54
DATA_LIST, 18
DEVICE, 120
DO_WHILE, 38
DYNAMIC_INCLUDE, 19
EDIT_CHANGED, 55
ELSE, 19
FILE, 122
FOCUS_IN, 48
FOR, 36
FORM, 19, 43
FUNCTION, 64
FUNCTION_BODY, 65
HELP_CONTEXT, 54
HMI_RESET, 19
IF, 20
IMG, 62
INCLUDE, 20
INDEX_CHANGED, 55
INIT, 45
KEY_EVENT, 46
LET, 21
MENU, 55
MESSAGE, 47
MSG, 24
MSGBOX, 25
NAME, 120
NAVIGATION, 56
NC_INSTRUCTION, 69
OP, 26
OPEN_FORM, 56
OPTION_MD, 124
PAINT, 48
PASSWORD, 31
PLC_INTERFACE, 124
POWER_OFF, 31, 125
PRINT, 32
PROGRESS_BAR, 33
PROPERTY, 57
REQUEST, 67
SEND_MESSAGE, 34, 48
SET_ACTIVE, 120
SET_INACTIVE, 120
SHOW_CONTROL, 35
SLEEP, 34
SOFTKEY, 58
SOFTKEY_ACCEPT, 61
SOFTKEY_BACK, 61
SOFTKEY_CANCEL, 60, 127
SOFTKEY_OK, 60, 127
START_UP, 120
STOP, 34
SWITCH, 35
TEST, 120
TEXT, 61
THEN, 35
TIMER, 48
TYPE_CAST, 35
UID, 120
UPDATE_CONTROLS, 67
VERSION, 120
WAITING, 36, 125
WHILE, 37
XML_PARSER, 27

SIEMENS


Introduction 1

Setting up OPC UA components 2

Testing the connection 3

OPC UA data access 4


# SINUMERIK

# SINUMERIK 840D sl/828D SINUMERIK Integrate for Engineering Access MyMachine / OPC UA

Commissioning Manual

Valid for:

CNC Software Version 4.7


06/2014
6FC5397-3DP40-5BA1

## Legal information

### Warning notice system

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



**Danger**

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



**Warning**

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



**Caution**

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.

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

Order number: 6FC5397-3DP40-5BA1
℗ 07/2014 Subject to change

Copyright © Siemens AG 2014.
All rights reserved

# Table of contents


**1 Introduction....5**

**2 Setting up OPC UA components....7**

2.1 Requirement....7

2.2 Setup....8

**3 Testing the connection....13**

**4 OPC UA data access....19**

4.1 Functional scope....20
4.1.1 Browsing....20
4.1.2 Variable paths....22
4.1.3 Supported services....26
4.1.4 Special features with regard to the OPC UA specification....27

**Index....29**

# Introduction 1

## Overview

OPC Unified Architecture (OPC UA) is a standard communication protocol for the industrial environment.



**Software option**

You require the following software option in order to use this function: SINUMERIK Integrate for Engineering "Access MyMachine / OPC UA".

Important features include:

- Independent of any particular platform
- Security
- Service call timeouts
- Heartbeat
- Buffering

Today, two protocols are available for transport via the network:

1. Binary protocol
   URL: opc.tcp://ServerTCP port 4840
   Optimized for
   - Low overhead
   - Low usage of resources
   - Interoperability
2. Web service (SOAP)
   URL: http://Server
   Ports 80 (http) and 443 (https)
   Optimized for
   - Tool support (used from "Java" or under ".NET")
   - Firewall compatibility

Information is provided in what is known as the "address space". This is comprised of "nodes" and "references". Detailed information on this is provided on the OPC Foundation (https://opcfoundation.org/) page.

The binary protocol is supported in the SINUMERIK environment. Currently, OPC UA Data Access is offered as service.

# Setting up OPC UA components

# 2

## 2.1 Requirement

The OPC UA components can be installed on PCU, NCU and PPU target systems.

The following steps are necessary to do this:

1. Extending the system configuration
2. Creating the OPC UA configuration file
3. Adapting the OPC UA configuration file
4. Setting the license
5. Enabling the communication port
6. Checking the HMI time
7. Performing a restart

# 2.2 Setup

## Extending the system configuration

In order that the OPC UA components start when SINUMERIK Operate runs up, the system configuration must be appropriately extended.

Copy the template file
".../siemens/sinumerik/hmi/template/cfg/MiniWeb_<Target system>_systemconfiguration.ini"
to
".../oem/sinumerik/hmi/cfg/systemconfiguration.ini".

**Note**

- The file name changes to "systemconfiguration.ini". The prefix "MiniWeb_<Target system>_" must be removed after copying.
- If the ".../oem/sinumerik/hmi/cfg/systemconfiguration.ini" file already exists, then you only have to copy the content of the template file.
- The template file is a link. Make sure that you copy the file and not only the link.

## Creating the OPC UA configuration file

Copy the template file
".../siemens/sinumerik/hmi/template/cfg/OPC_UAApplication.xml" to
".../oem/sinumerik/hmi/miniweb/WebCfg/OPC_UAApplication.xml".

**Note**

The template file is a link. Make sure that you copy the file and not only the link.

## Adapting the OPC UA configuration file

Open the copied file
".../oem/sinumerik/hmi/miniweb/WebCfg/OPC_UAApplication.xml" with a text editor.

Replace all the "localhost" entries in the file with the IPv4 address of the target system.

- For the NCU and PPU: -X130
- For the PCU 50: Local Area Connection 2

**Example**

```
<?xml version="1.0" standalone="yes"?>
<OPCUAAPPLICATION>
   <CERTDATA               KeyLength="1024"
                           Hash="SHA1"
                           txtC="DE"
                           txtP="Bavaria"
```

```
                              txtL="Erlangen"
                              txtO="Siemens AG"
                              txtOU="Industry"
                              txtCN=""
                              txtE="opcua@siemens.com"
                              txtCA="selfsigned" />
    <KEYS                     PublicKey="MiniWeb_OPCUA_certificate.crt"
                              PrivateKey="keys/MiniWeb_OPCUA_key.crt"
                              CAKey=""/>
    <SESSION                  MaxSessionCount="100"/>
    <BROWSE                   MaxNodesPerBrowse="50"/>
    <SUBSCRIPTION             MinSamplingRate="100"
                              MaxKeepAlive="20"
                              MinPublishRate="1000"/>
    <BUILDINFO                ProductName="Sinumerik OPC UA"
                              ProductUri="http://automation.siemens.com"
                              ManufacturerName="Siemens AG I DT MC"
                              SoftwareVersion="4.4.0"
                              BuildNumber="2"/>
    <APPLICATIONDESCRIPTION   ApplicationUri="urn:192.168.10.222:miniweb"
                              ApplicationNameLocale="en_us"
                              ApplicationNameText="Sinumerik OPC UA OEM"
                              DNSNAME="192.168.10.222"/>
    <ENDPOINTDESCRIPTION      URL="opc.tcp://192.168.10.222:4840"/>
    <NODEMANAGEMENT           TargetProviderName="NodeManagementProvider"/>
</OPCUAAPPLICATION>
```

## Setting the license

To access data with OPC UA clients, set the "Access MyMachine / OPC UA" license via the "Startup > Licenses" operating area.

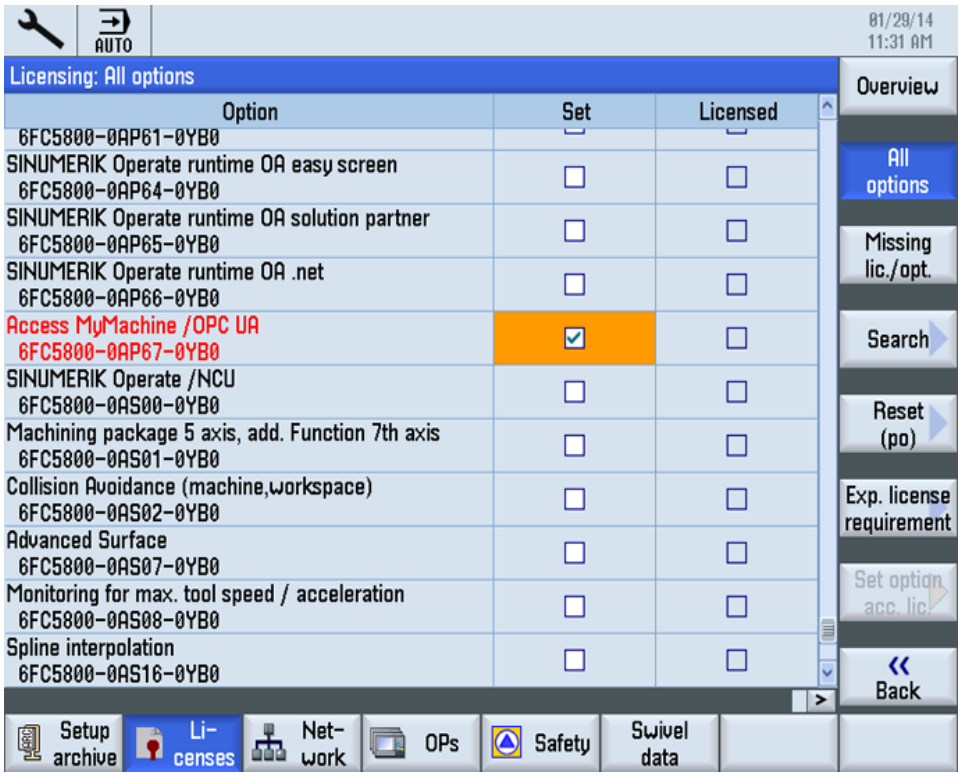



Figure 2-1 Setting the license

## Enabling the communication port

**Note**

Only perform this step on the NCU and PPU target systems. Go directly to the next step on the PCU 50.

Enable the OPC UA communication port (TCP port 4840) via the "Startup > Network > Company network" operating area.

Press the "Change" button.

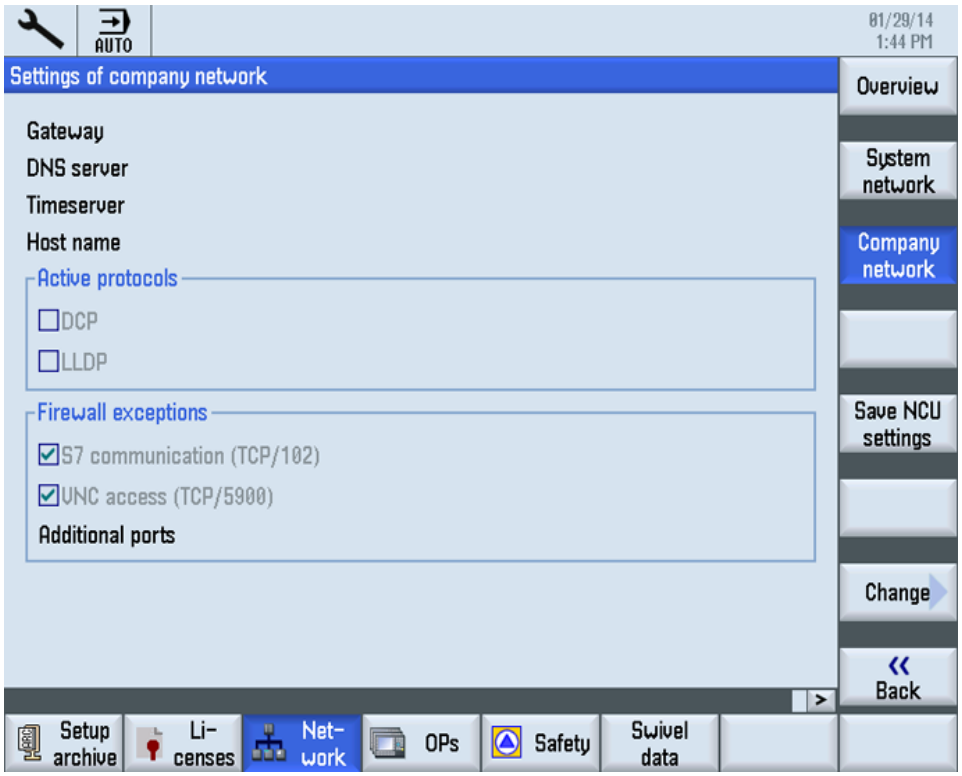



Figure 2-2 Enabling the communication port

Add the TCP port 4840 and confirm the changes with "OK".

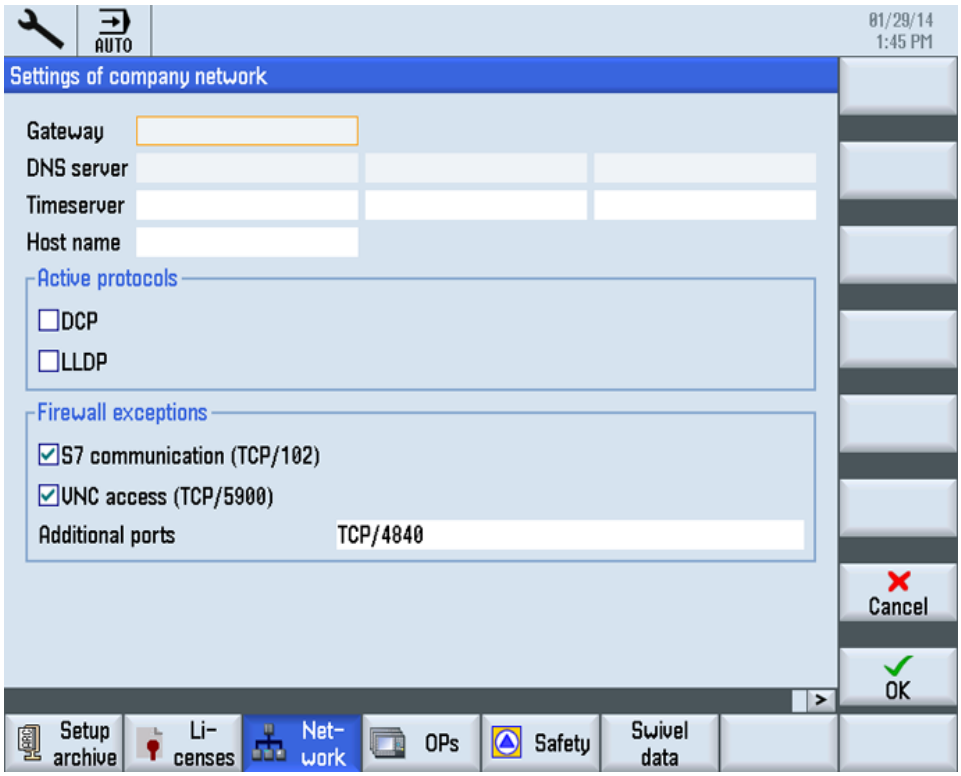



Figure 2-3 Adding the TCP port

## Checking the HMI time

You must correctly set the HMI time so that the secure OPC UA communication functions.

**Note**

The certificate required for the secure OPC UA communication (../siemens/sinumerik/hmi/miniweb/System/SSL/MiniWeb_OPCUA_certificate.crt) is generated automatically at the first run-up. The start of validity is set to the current time. The validity period is 30 years.

If the time is subsequently changed so that it lies outside the validity period, the secure OPC UA communication does not function.

## Performing a restart

Perform a restart of the SINUMERIK Operate and a reset (po) to activate the changes and the license.

# Testing the connection

# 3

## Requirement

You can use the "Sample client" of the OPC Foundation (http://www.opcf.org/) at "Downloads / Sample Code and Apps / OPC UA 1.02 Sample Applications" to test the connection.

**Note**

There are two ways to establish the connection:

- Connection without security
- Connection with the security policy "Basic128Rsa15" and the security mode "SignAndEncrypt".

## Procedure

1. Start the OPC UA "Sample client".

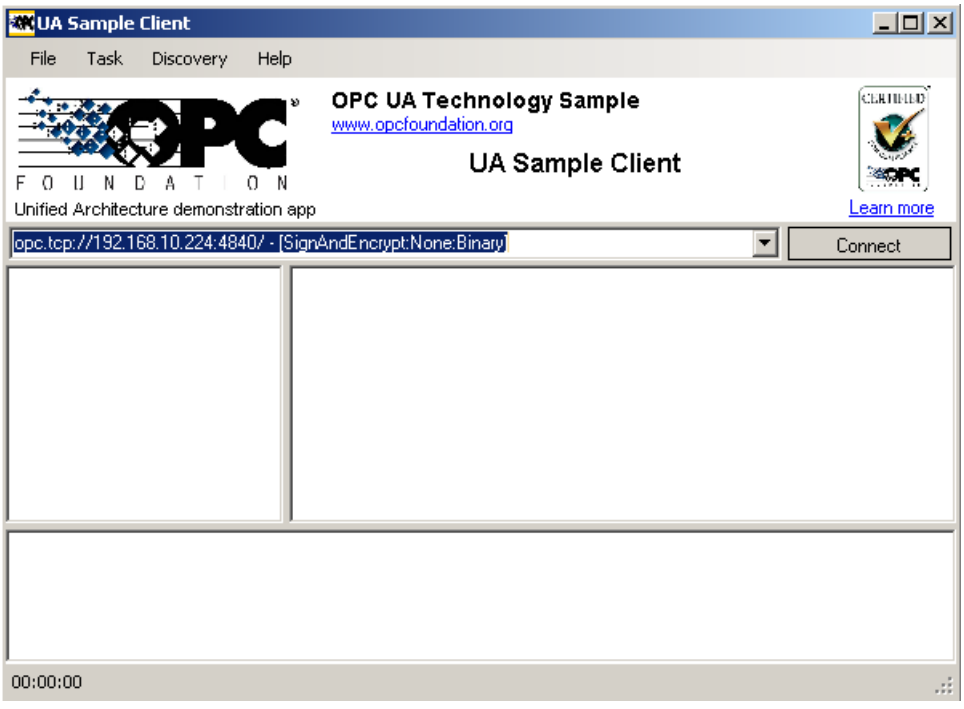



Figure 3-1 Sample Client main window

2. Select the "New" entry from the drop-down list.
   The "Discover Servers" window opens.

3. Now enter the IPv4 address of the target system and click the "Discover" button.

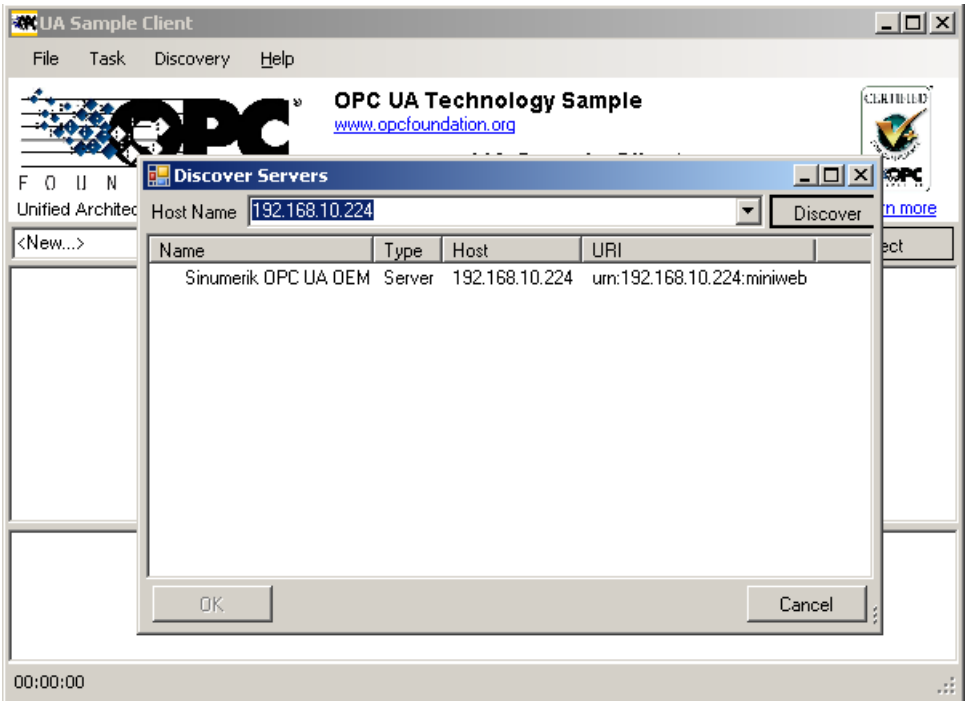



Figure 3-2 Discover servers

4. The SINUMERIK OPC UA server appears in the list. Select the server and confirm with "OK".
5. Return to the main window and click the "Connect" button.

6. To establish a simple connection without security, make the following settings and confirm the settings with "OK".

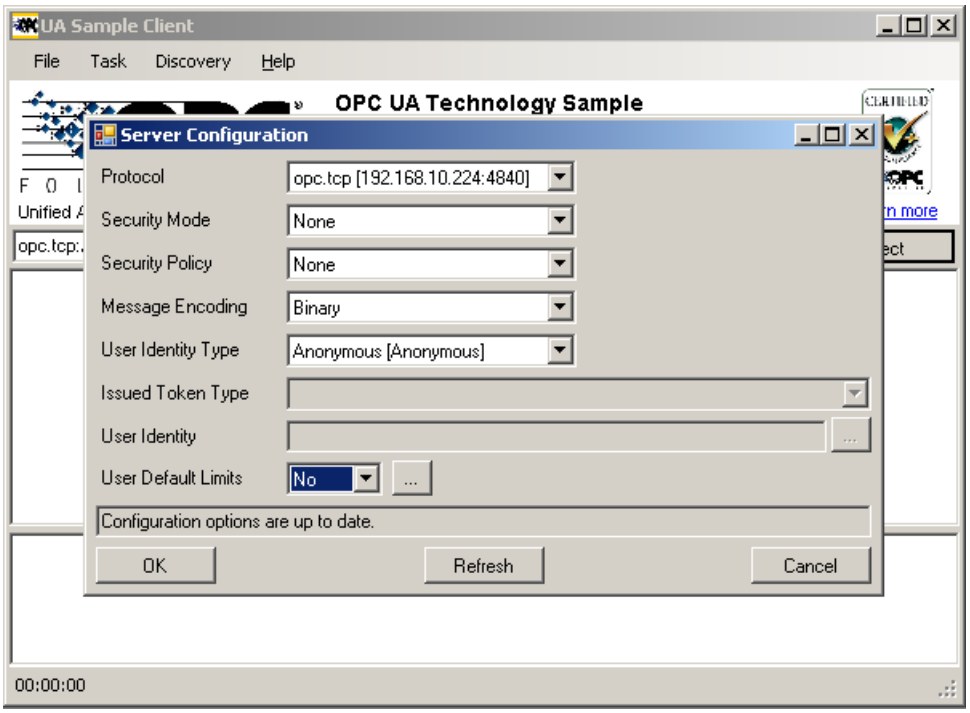



Figure 3-3 Server configuration

7. Confirm the following "Open Session" dialog with "OK".

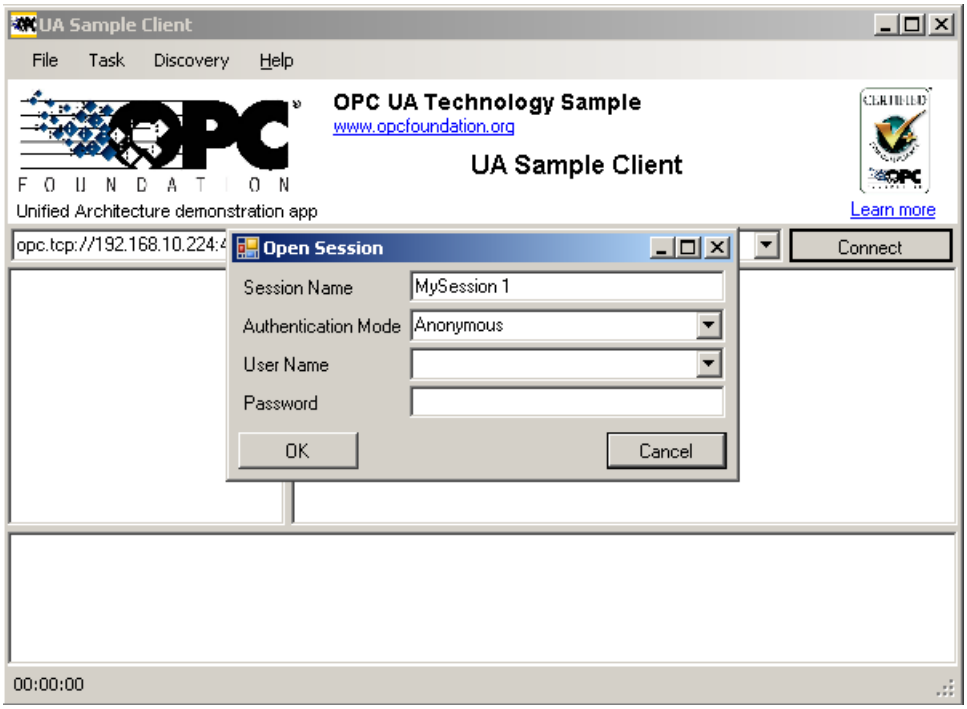



Figure 3-4 Open session

The connection to the SINUMERIK OPC UA server has now been established and the available address space is displayed.

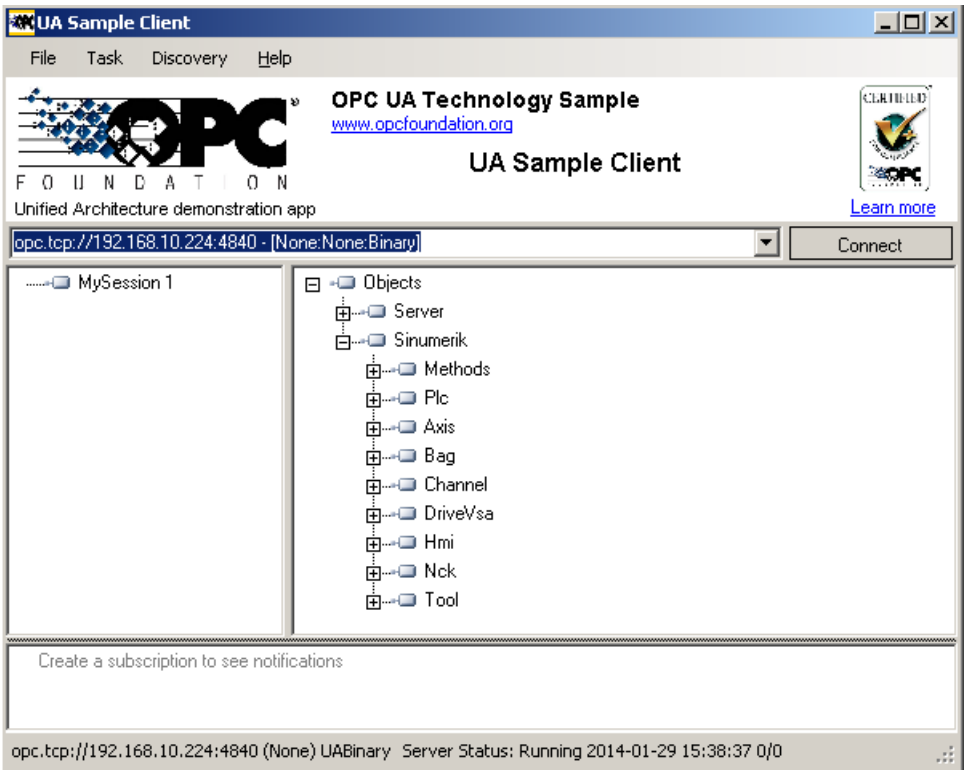



Figure 3-5 Address space of the SINUMERIK OPC UA server

8. Now navigate to a nodeID (e.g. R-parameter at Sinumerik > Channel > Parameter > R) and right-click the corresponding entry. You can now test various functions:
   – E.g. read, write, setup monitoring

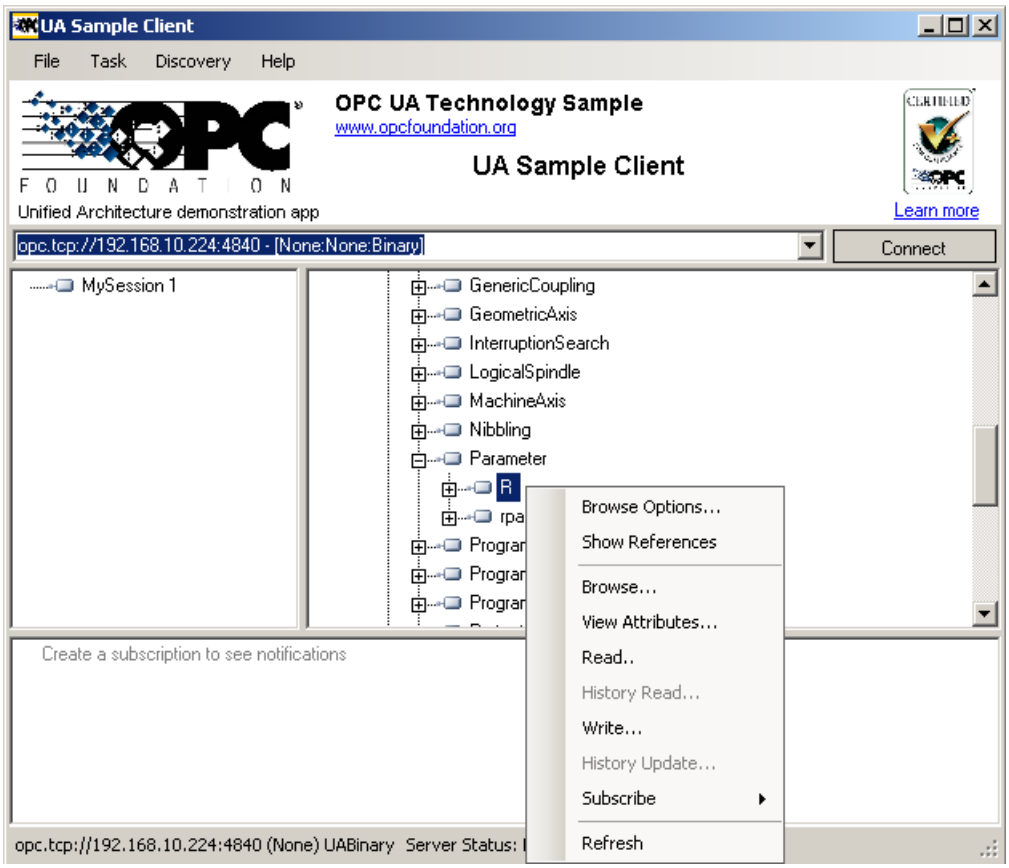



Figure 3-6 NodeID "Sinumerik > Channel > Parameter > R"

– The attributes of a nodeID can be viewed via the "View Attributes" entry. One of these attributes is "Value", which supplies the appropriate value of R1.

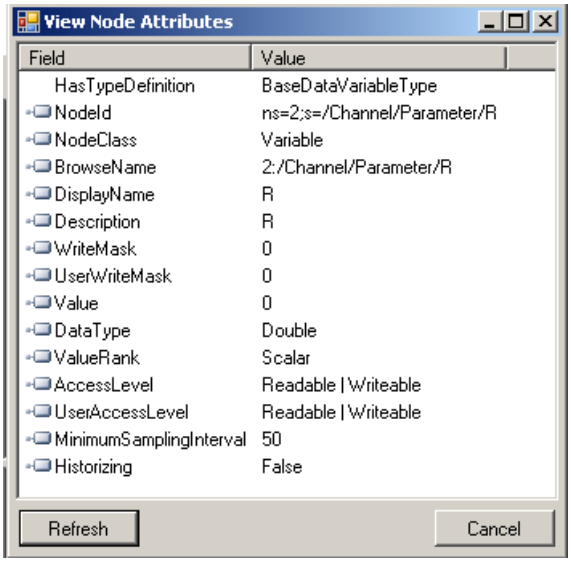



Figure 3-7 Viewing node attributes

# OPC UA data access

# 4

## Objective of the section

This section describes the functional scope and the SINUMERIK-specific special features of the OPC UA data access interface.

**Note**

The communication time is not guaranteed. As a consequence, realtime tasks cannot be implemented using the OPC UA data access interface.

Detailed information on the OPC UA specification is provided on the OPC Foundation (https://opcfoundation.org/) page.

# 4.1 Functional scope

Using the OPC UA data access interface, data can be accessed on the NC and PLC. In particular, variables can be read and written, and you can be notified when a value changes.

You can access individual variables (e.g. /Channel/Parameter/R[u1,1]) and array variables (e.g. /Channel/Parameter/R[u1,1,10]).

In addition, the following two methods are available for reading and writing:

- Sinumerik/Methods/ReadVar
- Sinumerik/Methods/WriteVar

**Note**

**Limited number of simultaneously monitored variables**

The number of variables from the "Sinumerik" object that can be monitored simultaneously is limited:

- SINUMERIK 828D, maximum 20 variables
- SINUMERIK 840D sl, maximum 200 variables

If you attempt to monitor more variables, the "BadWaitingForInitialData" error is returned.

## 4.1.1 Browsing

If the OPC UA data access interface is browsed, then the available address space is mapped under "Sinumerik".

Global User Data (GUD) can be found under the "/Sinumerik/GUD" node.

The PLC blocks (inputs, outputs, bit memory, data blocks) can be found under the "/Sinumerik/Plc" node.

Observe the following during browsing:

- In the address space of the NC, the displayed variables always represent only the first parameter of the corresponding first OPI unit.
  Example: The R parameters are found at "Sinumerik > Channel > Parameter > R". The corresponding identifier is called: "/Channel/Parameter/R", which is eventually mapped to "/Channel/Parameter/R[u1, 1]". If you want to access other parameters, then you must correspondingly extend the identifier; you cannot directly accept the identifiers obtained when browsing, e.g. "/Channel/Parameter/R[u2, 56]".
- In the address space of the NC, the displayed variables represent the access format that has to be extended accordingly.
  Example: The variable "/Plc/MB" is in the address space. This variable must be extended by the appropriate byte number, e.g. "/Plc/MB6". Only then is a value supplied.

- The address space of the NC also contains variables that are not available in a corresponding machine configuration. These variables return "BadAttributeIdInvalid" as value.
- The address space of the machine data and setting data cannot be accessed via browsing. How access is possible and how the identifiers are composed, is explained in SectionVariable paths (Page 22).

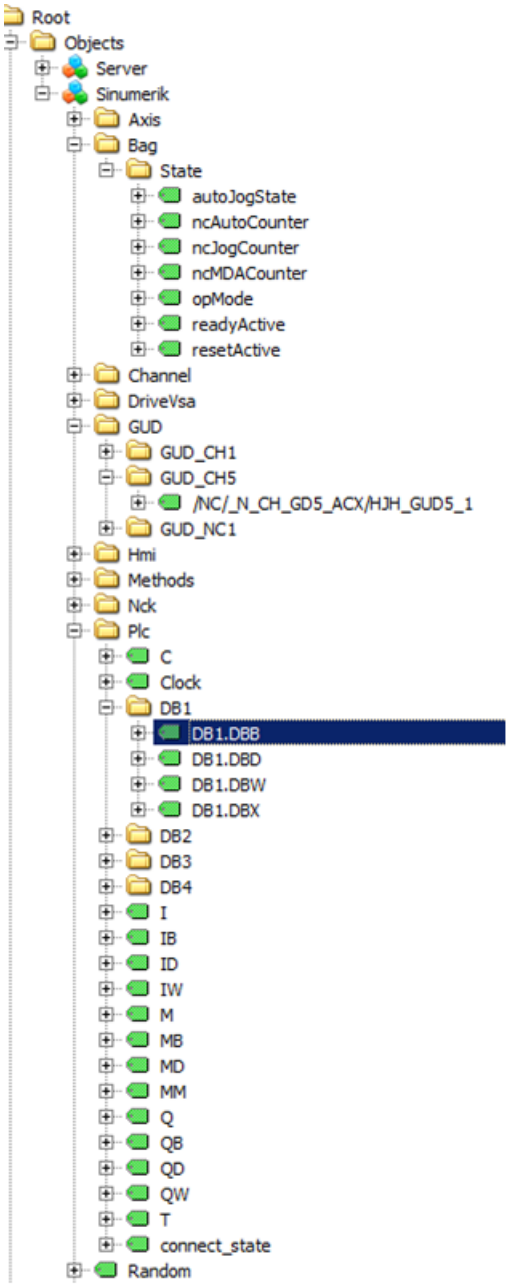



Figure 4-1 Excerpt from browsing the OPC UA data access interface

## 4.1.2 Variable paths

### Variable paths for NC access operations

The variable paths for NC access are stored in the address space of the SINUMERIK Operate OPC UA server. Pay attention to the correct upper-case and lower-case of the "nodeID". The respective identifier of the "nodeID" provides information on the correct notation.

You can obtain additional information on this from the List Manuals for 840D sl and 828D sl "NC variable and interface signals".

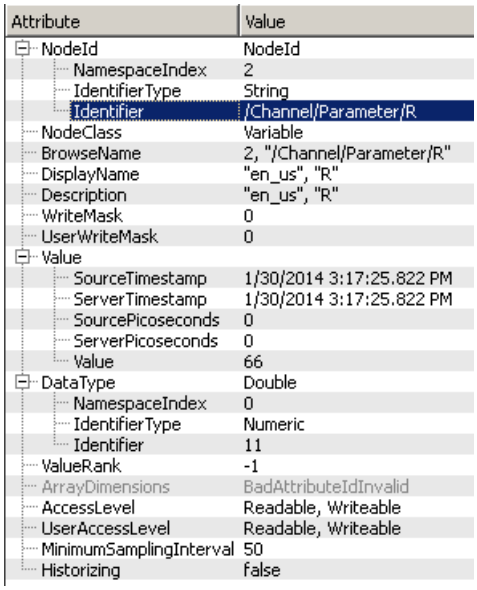



Figure 4-2 Identifier for R parameter

The displayed NC variables always represent only the first parameter of the corresponding NC data area (channel, TO area, mode group).

#### Example

The R parameters are found under the identifier "/Channel/Parameter/R", which is eventually mapped to "/Channel/Parameter/R[u1, 1]". If you want to access other parameters, then you must correspondingly extend the identifier, e.g. "/Channel/Parameter/R[u2, 56]".

Table 4-1 Examples of variable paths (NC accesses)

| Variable path | Description |
|---|---|
| /Channel/Parameter/R[u1,10] | R parameter 10 in channel 1 |
| /Channel/Parameter/R[u1,1,5] | R parameter array |
| /Channel/Parameter/R[u1,1,#5] | R parameters 1 to 5 in channel 1 |
| /Channel/GeometricAxis/name[u2,3] | Name of the 3rd axis in channel 2 |
| /Channel/GeometricAxis/actToolBasePos[u1,3] | Position of the 3rd axis in channel 1 |

## Variable paths for GUD accesses

The variable paths for NC access are stored in the address space of the SINUMERIK Operate OPC UA server under the "/Sinumerik/GUD" node. Pay attention to the correct upper-case and lower-case of the "nodeID". The respective identifier of the "nodeID" provides information on the correct notation.

The displayed GUD variables always represent only the first parameter (for GUD arrays) of the first NC channel (for channel-dependent GUD variables). If you want to access a different parameter of a GUD array or a different channel, you must extend the identifier accordingly for the NC access.

GUD arrays are 1-indexed for access, and access is always one-dimensional. This means, the index must be calculated for multi-dimensional arrays.

### Example 1: One-dimensional array, NC-global GUD array

"UGUD.DEF" file

```
DEF NCK INT ARRAY[2]
M17
```

Access is performed as follows:

```
ARRAY[0] → /NC/_N_NC_GD3_ACX/ARRAY[1]
ARRAY[1] → /NC/_N_NC_GD3_ACX/ARRAY[2]
```

### Example 2: Two-dimensional array, channel-dependent GUD array

"UGUD.DEF" file

```
DEF CHAN INT ABC[3,3]
M17
```

Access is performed as follows:

```
ABC[0,0] → /NC/_N_CH_GD3_ACX/ABC[u1, 1]
ABC[0.1] → /NC/_N_CH_GD3_ACX/ABC[u1, 2]
ABC[0.2] → /NC/_N_CH_GD3_ACX/ABC[u1, 3]
ABC[1.0] → /NC/_N_CH_GD3_ACX/ABC[u1, 4]
ABC[1.1] → /NC/_N_CH_GD3_ACX/ABC[u1, 5]
ABC[1.2] → /NC/_N_CH_GD3_ACX/ABC[u1, 6]
ABC[2.0] → /NC/_N_CH_GD3_ACX/ABC[u1, 7]
ABC[2.1] → /NC/_N_CH_GD3_ACX/ABC[u1, 8]
ABC[2.2] → /NC/_N_CH_GD3_ACX/ABC[u1, 9]
```

## Variable paths for PLC access operations

The variable paths for PLC access are stored in the address space of the SINUMERIK Operate OPC UA server under the "/Sinumerik/PLC" node. Pay attention to the correct upper-case and lower-case of the "nodeID". The respective identifier of the "nodeID" provides information on the correct notation.

In the address space of the NC, the displayed variables represent the access format that has to be extended accordingly.

### Example

The variable "/Plc/MB" is in the address space. This variable must be extended by the appropriate byte number, e.g. to "/Plc/MB6". Only then is a value supplied.

The various access formats are shown in the following table. These access formats must be extended by the prefix "/Plc".

**Note**

The data type is converted during access with the OPC UA data access interface. Refer to the following table for the data type conversions.

Table 4-2 PLC syntax

| Area | Address (IEC) | Permissible data types | OPC UA data type |
|---|---|---|---|
| Output image | Qx.y | **BOOL** | Boolean |
| Output image | QBx | **BYTE**, CHAR, STRING | UInt32 String |
| Output image | QWx | **WORD**, CHAR, INT, | UInt32 Int32 |
| Output image | QDx | **DWORD**, DINT, REAL | UInt32 Int32 Double |
| Data block | DBz.DBXx.y | **BOOL** | Boolean |
| Data block | DBz.DBBx | **BYTE**, CHAR, STRING | UInt32 String |
| Data block | DBz.DBWx | **WORD**, CHAR, INT | UInt32 Int32 |
| Data block | DBz.DBDx | **DWORD**, DINT, REAL | UInt32 Int32 Double |
| Input image | Ix.y | **BOOL** | Boolean |
| Input image | IBx | **BYTE**, CHAR, STRING | UInt32 String |
| Input image | IWx | **WORD**, CHAR, INT | UInt32 Int32 |
| Input image | IDx | **DWORD**, DINT, REAL | UInt32 Int32 Double |
| Bit memory | Mx.y | **BOOL** | Boolean |
| Bit memory | MBx | **BYTE**, CHAR, STRING | UInt32 String |
| Bit memory | MWx | **WORD**, CHAR, INT | UInt32 Int32 |

| Area | Address (IEC) | Permissible data types | OPC UA data type |
|---|---|---|---|
| Bit memory | MDx | **DWORD**,<br>DINT,<br>REAL | UInt32<br>Int32<br>Double |
| Counters | Cx | **WORD** | UInt32 |
| timers | Tx | - | Int32 |

Notes regarding the table:

- In the table, "x" stands for the byte offset in the area; "y" for the bit number in the byte; and "z" for the data block.
- The **bold data type** is the default data type in each case and does not have to be specified when addressing. In addition, the specifications DB2.DBB5.BYTE and DB2.DBB5 are equivalent, for example.
- Square brackets are used to access arrays, e.g. "/Plc/DB5.DBW2:[10]" (word array of length 10).

Table 4-3 Examples of variable paths (PLC accesses)

| Variable path | Description |
|---|---|
| /Plc/M5.0 | Memory bit 0 at byte offset 5 |
| /Plc/DB5.DBW2 | Word (16-bit) at byte offset 2 in data block 5 |
| /Plc/DB8.DBB2:STRING | UTF8 string beginning at byte offset 2 in data block 8 |
| /Plc/DB8.DBW2:[10] | Array of 10 words beginning at byte offset 2 in data block 8 |
| /Plc/DB100.DBB1 | Byte at byte offset 1 in data block 100 |
| /Plc/DB2.DBD0:REAL[10] | Array of 10 double words (32-bit) beginning at byte offset 0 in data block 2, which are formatted as a floating-point number |

Note the following:

**Note**

- Timers can only be read. A timer is active if it contains a value other than 0.
- If the data type CHAR or STRING is used in conjunction with a byte access, UTF8 characters are read, but if either data type is used in conjunction with a word access, UTF16 characters are read.
- Variables of the STRING type contain the maximum length in the first byte and the actual length in the second byte. When strings are written, the actual length is adapted accordingly. The maximum length is not changed.
- For the STRING data type in conjunction with a byte access (e.g. "/Plc/DB99.DBB0:STRING"), the maximum string length is 255 characters. As a result of the UTF8 formatting, for some characters (e.g. for the "µ"), two bytes are required, so that the maximum string length is correspondingly reduced.
- Only one-dimensional arrays are supported. With STRING arrays, all elements have the same maximum length.

## Variable paths for machine and setting data

The variable paths of the machine data and setting data cannot be accessed via browsing.

The variable path for reading, writing and monitoring machine data and setting data comprises a defined prefix and the machine data or setting data name.

The following prefixes are possible:

Table 4-4 Prefixes for machine and setting data

| Area | Prefix |
|---|---|
| General machine data | /Random/NC/_N_NC_TEA_ACX |
| Channel-specific machine data | /Random/NC/_N_CH_TEA_ACX |
| Axis-specific machine data | /Random/NC/_N_AX_TEA_ACX |
| General setting data | /Random/NC/_N_NC_SEA_ACX |
| Channel-specific setting data | /Random/NC/_N_CH_SEA_ACX |
| Axis-specific setting data | /Random/NC/_N_AX_SEA_ACX |

Machine data arrays are 1-indexed for access.

Table 4-5 Examples of variable paths (machine and setting data access operations)

| Variable path | Description |
|---|---|
| /Random/NC/_N_NC_TEA_ACX/$MN_IPO_CYCLE_TIME | IPO cycle clock (general MD) |
| /Random/NC/_N_CH_TEA_ACX/$MC_DISPLAY_AXIS[u2, 1] | Display axis on HMI (channel-specific MD) |
| /Random/NC/_N_AX_SEA_ACX/$SA_LEAD_TYPE[u2] | Type of master value (axis-specific SD) |

### 4.1.3 Supported services

The following services of the OPC UA data access specification are supported:

- ActivateSessionService
- AddNodesService
- AddReferencesService
- BrowseNextService
- BrowseService
- CallService
- CloseSessionService
- CreateMonitoredItemsService
- CreateSessionService
- CreateSubscriptionService
- DeleteMonitoredItemsService
- DeleteNodesService

- DeleteReferencesService
- DeleteSubscriptionService
- FindServersService
- GetEndpointsService
- ModifyMonitoredItemsService
- PublishService
- ReadService
- RegisterNodesService
- RepublishService
- SetMonitoringModeService
- SetPublishingModeService
- TransferSubscriptionsService
- TranslateBrowsePathsToNodeIdsService
- UnregisterNodesService
- WriteService

### 4.1.4 Special features with regard to the OPC UA specification

#### TranslateBrowsePathsToNodeIdsService

If a non-existing nodeID is specified in TranslateBrowsePathsToNodeIds, then "BadNoMatch" is returned instead of BadNodeIdUnknown".

# Index


## A

Architecture, 5

## B

BadWaitingForInitialData, 20
Binary protocol, 5

## C

Communication port, 10

## D

Data types, 24

## L

License, 10

## M

Method
    ReadVar, 20
    WriteVar, 20

## O

OPC_UAApplication.xml, 8

## R

ReadVar, 20

## W

WriteVar, 20

SIEMENS

# SINUMERIK

# SINUMERIK 840D sl/828D
# SINUMERIK Integrate Run
# MyScreens

Programming Manual


Introduction 1

Getting Started 2

Fundamentals 3

Dialogs 4

Variables 5

Programming commands 6

Graphic and logic elements 7

"Custom" operating area 8

Dialog selection 9

Reference lists A

Tips and tricks B

Animated elements C

Glossary 10


Valid for:

CNC software V4.7 SP1


01/2015
6FC5397-3DP40-5BA2

## Legal information

### Warning notice system

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

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

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

| ⚠ CAUTION |
| --- |
| 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.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

Order number: 6FC5397-3DP40-5BA2
℗ 02/2015 Subject to change

Copyright © Siemens AG 2014 - 2015.
All rights reserved

# Table of contents


**1 Introduction....9**

**2 Getting Started....13**

2.1 Introduction....13

2.2 Example....13
2.2.1 Task description....13
2.2.2 Creating the configuration file....17
2.2.3 Saving the configuration file in the OEM directory....20
2.2.4 Creating the online help....21
2.2.5 Saving the online help in the OEM directory....23
2.2.6 Copying easyscreen.ini into the OEM directory....23
2.2.7 Registering the COM file in easyscreen.ini....24
2.2.8 Testing the project....24

**3 Fundamentals....25**

3.1 Structure of configuration file....25

3.2 Structure of the menu tree....27

3.3 Definition and functions for start softkeys....28
3.3.1 Defining the start softkey....28
3.3.2 Functions for start softkeys....29

3.4 Troubleshooting (log book)....31

3.5 Notes on the easyscreen.ini....32

3.6 Notes for personnel changing over to "Run MyScreens"....34

3.7 Extended configuration syntax....36

**4 Dialogs....39**

4.1 Structure and elements of a dialog....39
4.1.1 Defining a dialog....39
4.1.2 Defining dialog properties....41
4.1.3 Defining dialog elements....45
4.1.4 Defining dialogs with multiple columns....47
4.1.5 Password dialogs....48
4.1.6 Using display images/graphics....50

4.2 Defining softkey menus....50
4.2.1 Changing softkey properties during runtime....53
4.2.2 Language-dependent text....56

4.3 Configuring the online help....58

**5 Variables....61**

5.1 Defining variables....61

5.2 Application examples....62

5.3 Example 1: Assigning the variable type, texts, help display, colors, tooltips....63
5.4 Example 2: Assigning the Variable Type, Limits, Attributes, Short Text Position properties....64
5.5 Example 3: Assigning the Variable Type, Default, System or User Variable, Input/Output Field Position properties....65
5.6 Example 4: Toggle field and list field....65
5.7 Example 5: Image display....66
5.8 Example 6: Progress bar....66
5.9 Example 7: Password input mode (asterisk)....69
5.10 Variable parameters....69
5.11 Details on the variable type....75
5.12 Details on the toggle field....78
5.13 Details on the default setting....79
5.14 Details on the position of the short text, position of the input/output field....80
5.15 Use of strings....80
5.16 CURPOS variable....82
5.17 CURVER variable....82
5.18 ENTRY variable....83
5.19 ERR variable....83
5.20 FILE_ERR variable....84
5.21 FOC variable....85
5.22 Variable S_ALEVEL....86
5.23 S_CHAN variable....87
5.24 Variable S_CONTROL....87
5.25 Variable S_LANG....88
5.26 Variable S_NCCODEREADONLY....89
5.27 Variables S_RESX and S_RESY....89
**6 Programming commands....91**
6.1 Operators....91
6.1.1 Mathematical operators....91
6.1.2 Bit operators....94
6.2 Methods....95
6.2.1 ACCESSLEVEL....96
6.2.2 CHANGE....96
6.2.3 CHANNEL....98
6.2.4 CONTROL....98
6.2.5 FOCUS....99
6.2.6 LANGUAGE....100

6.2.7 LOAD....100
6.2.8 UNLOAD....101
6.2.9 OUTPUT....102
6.2.10 PRESS....103
6.2.11 PRESS(ENTER)....104
6.2.12 PRESS(TOGGLE)....104
6.2.13 RESOLUTION....105
6.2.14 RESUME....105
6.2.15 SUSPEND....106
6.2.16 Example: Version management with OUTPUT methods....106

6.3 Functions....108
6.3.1 Reading and writing drive parameters: RDOP, WDOP, MRDOP....108
6.3.2 Subprogram call (CALL)....110
6.3.3 Define block (//B)....111
6.3.4 Check Variable (CVAR)....112
6.3.5 CLEAR_BACKGROUND....113
6.3.6 Copy Program file function (CP)....114
6.3.7 Delete Program file function (DP)....114
6.3.8 Exist Program file function (EP)....115
6.3.9 Move Program file function (MP)....116
6.3.10 Select Program file function (SP)....117
6.3.11 File accesses: RDFILE, WRFILE, RDLINEFILE, WRLINEFILE....118
6.3.12 Dialog line (DLGL)....120
6.3.13 DEBUG....121
6.3.14 Exit dialog (EXIT)....122
6.3.15 Dynamic manipulation of the lists of toggle fields or list box fields....123
6.3.16 Evaluate (EVAL)....126
6.3.17 Exit Loading Softkey (EXITLS)....127
6.3.18 Function (FCT)....127
6.3.19 Generate code (GC)....129
6.3.20 Password functions....132
6.3.21 Load Array (LA)....133
6.3.22 Load Block (LB)....134
6.3.23 Load Mask (LM)....135
6.3.24 Load Softkey (LS)....136
6.3.25 Load Grid (LG)....137
6.3.26 Multiple selection SWITCH....138
6.3.27 Multiple Read NC PLC (MRNP)....139
6.3.28 PI services....141
6.3.29 Reading (RNP) and writing (WNP) system or user variables....142
6.3.30 RESIZE_VAR_IO and RESIZE_VAR_TXT....143
6.3.31 Register (REG)....144
6.3.32 RETURN....145
6.3.33 Recompile....146
6.3.34 Recompile without user comment....147
6.3.35 Search Forward, Search Backward (SF, SB)....150
6.3.36 STRING functions....151
6.3.37 WHILE/UNTIL loops....157
6.3.38 Cyclic execution of scripts: START_TIMER, STOP_TIMER....160

**7 Graphic and logic elements....163**

7.1 Line, dividing line, rectangle, circle and ellipse....163

7.2 Defining an array....165
7.2.1 Accessing the value of an array element....166
7.2.2 Example Access to an array element....168
7.2.3 Scanning the status of an array element....170
7.3 Table description (grid)....170
7.3.1 Defining a table (grid)....172
7.3.2 Defining columns....172
7.3.3 Focus control in the table (cell)....174
7.4 Custom widgets....175
7.4.1 Defining custom widgets....175
7.4.2 Structure of the custom widget library....175
7.4.3 Structure of the custom widget interface....176
7.4.4 Interaction between custom widget and dialog box - Automatic data exchange....178
7.4.5 Interaction between custom widget and dialog box - Manual data exchange....179
7.4.5.1 Reading and writing properties....179
7.4.5.2 Executing a method of the custom widget....180
7.4.5.3 Response to a custom widget signal....182
**8 "Custom" operating area....185**
8.1 How to activate the "Custom" operating area....185
8.2 How to configure the "Custom" softkey....185
8.3 How to configure the "Custom" operating area....186
8.4 Programming example for the "Custom" area....188
**9 Dialog selection....193**
9.1 Dialog selection using PLC softkeys....193
9.2 Dialog selection using PLC hard keys....194
9.3 Dialog selection via NC....197
**A Reference lists....199**
A.1 Lists of start softkeys....199
A.1.1 List of start softkeys for turning....199
A.1.2 List of start softkeys for milling....200
A.2 List of access levels....202
A.3 List of colors....202
A.4 List of language codes used in file names....203
A.5 List of accessible system variables....204
A.6 Behavior when opening the dialog (attribute CB)....204
**B Tips and tricks....207**
B.1 General tips....207
B.2 Tips for debugging....208
B.3 Tips for the CHANGE method....208
B.4 Tips for DO LOOP loops....209

**C Animated elements....211**

C.1 Introduction....211

C.2 Modeling....212
C.2.1 Requirements....212
C.2.2 Rules for modeling....212
C.2.3 Importing graphics (models)....215
C.2.4 Modeling templates....217

C.3 XML commands....218
C.3.1 Overview....218
C.3.2 Structure of the scene description file....219
C.3.3 Mirroring and rotations....222
C.3.4 View type....222

C.4 Conversion to hmi file....223

C.5 Display in Create MyHMI /3GL....223
C.5.1 X3D Viewer....223
C.5.2 Class SIX3dViewerWidget....223
C.5.3 Public methods....224
C.5.4 Public slots....224
C.5.5 Libraries....225
C.5.6 Implementation example....225
C.5.7 Machine data....225
C.5.8 Notes about use....226

C.6 Display in Run MyScreens....226

**10 Glossary....229**

**Index....233**

# Introduction 1

## Overview

"Run MyScreens" can be used to create user interfaces that display functional expansions designed by the machine manufacturer or user, or to implement your own layout.

Cycle calls can also be generated with user interfaces that you have created. Dialogs can be created directly on the control system.

"Run MyScreens" is implemented with an Interpreter and configuration files that describe the user interfaces.

"Run MyScreens" is configured using ASCII files: These configuration files contain the description of the user interface. The syntax that must be applied in creating these files is described in the following chapters.

## Basic configuration

The "Run MyScreens" function enables machine manufacturers to configure their own dialogs. Even with the basic configuration, it is possible to configure five dialogs in the operator menu tree or for customer-specific cycle dialogs.



**Software option**

To expand the number of dialogs, you require one of the following software options:

- SINUMERIK 828D/840D sl, SINUMERIK Integrate Run MyScreens (6FC5800-0AP64-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyScreens + Run MyHMI (6FC5800-0AP65-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyHMI / 3GL (6FC5800-0AP60-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyHMI / WinCC (6FC5800-0AP61-0YB0)

## Supplementary conditions

The following conditions must be met:

- It is only possible to switch between dialogs within a single operating area.
- User variables may not have the same names as system or PLC variables (also see List Manual System Variables /PGAsl/).
- The dialogs activated by the PLC form a separate operating area (similar to measuring cycle screens).

## Tools

- UTF8-capable editor (e.g. the integrated editor of the user interface or Notepad)
- A graphics program is needed to create graphics/display images.

## File names and coding

**Note**

When using HMI Operate in the NCU, note that all file names are written in lower case letters (com, png, txt) on the CF card. This is required because of Linux.

On the PCU, you can write file names either in upper case or lower case letters. However, it is recommended that you also use lower case letters here, in case of a possible subsequent transfer to Linux.

**Note**

Make sure that the coding is set to UTF-8 in the editor that you are using when saving the configuration and language files.

## Storage paths

Note the following convention when storing the configuration files, language files, etc.

| | | |
|---|---|---|
| *[System Siemens directory]* | | |
| | Linux: | /card/siemens/sinumerik/hmi |
| | Windows 7: | C:\ProgramData\Siemens\MotionControl\siemens |
| *[System oem directory]* | | |
| | Linux: | /card/oem/sinumerik/hmi |
| | Windows 7: | C:\ProgramData\Siemens\MotionControl\oem |
| *[System user directory]* | | |
| | Linux: | /card/user/sinumerik/hmi |
| | Windows 7: | C:\ProgramData\Siemens\MotionControl\user |
| *[System add_on directory]* | | |
| | Linux: | /card/addon/sinumerik/hmi |
| | Windows 7: | C:\ProgramData\Siemens\MotionControl\addon |
| | | |
| Storage locations | | |
| | "Run MyScreens" configuration: | *[System oem directory]*/proj |
| | Configuration file: | *[System oem directory]*/cfg |
| | Language files: | *[System oem directory]*/lng |

| | | |
|---|---|---|
| | Image files: | *[System oem directory]*/lco/ico*[Resolution]*<br>Example:<br>*[System oem directory]*/lco/ico640 |
| | Online Help: | *[System oem directory]*/hlp/*[Language]*<br>Example:<br>*[System oem directory]*/hlp/eng |

## Application

You can implement the following functions:

| | |
|---|---|
| Display dialogs containing the following elements: | • Softkeys<br>• Variables<br>• Texts and Help texts<br>• Graphics and Help displays |
| Open dialogs by: | • Pressing the (start) softkeys<br>• Selection from the PLC/NC |
| Restructure dialogs dynamically: | • Edit and delete softkeys<br>• Define and design variable fields<br>• Insert, exchange and delete display texts (language-dependent or independent)<br>• Insert, exchange and delete graphics |
| Initiate operations in response to the following actions: | • Displaying dialogs<br>• Input values (variables)<br>• Select a softkey<br>• Exiting dialogs<br>• Value change of a system or user variable |
| Data exchange between dialogs: | |
| Variables: | • Read (NC, PLC and user variables)<br>• Write (NC, PLC and user variables)<br>• Combine with mathematical, comparison or logic operators |
| Execute functions: | • Subprograms<br>• File functions<br>• PI services |
| Apply protection levels according to user classes | |

# Getting Started

# 2

## 2.1 Introduction

Using the following example, you get to know the steps necessary to insert your own dialogs into the SINUMERIK Operate user interface using Run MySreens. You also learn how to create your own dialogs, insert context-sensitive help screens and help calls, define softkeys and how you can navigate between the dialogs.

## 2.2 Example

### 2.2.1 Task description

You will create the following dialogs in the example.

#### Dialog 1

R parameters that can be written to (0 and 1) and geometry axis names (input fields) are displayed in the first dialog. The corresponding help screens are linked in for the two R parameters. A context-sensitive help is linked in for the geometry axes. The dialog also contains examples of dividing lines (horizontal and vertical), toggle fields, I/O fields with integrated unit selection and for progress bars (with and without color change).

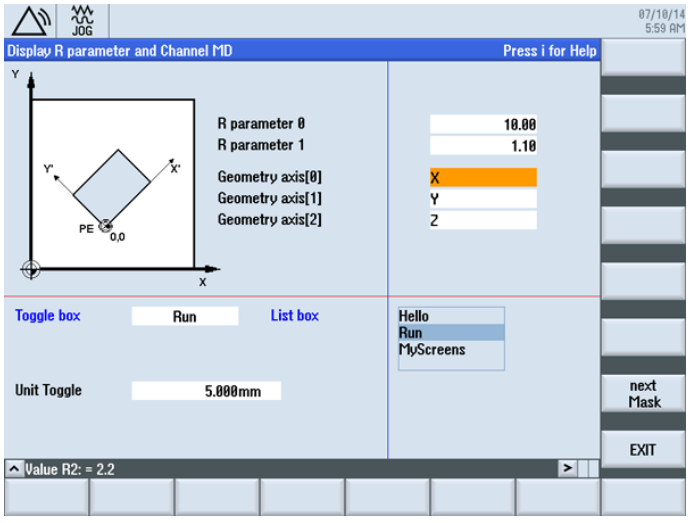



Figure 2-1 R parameters with help displays

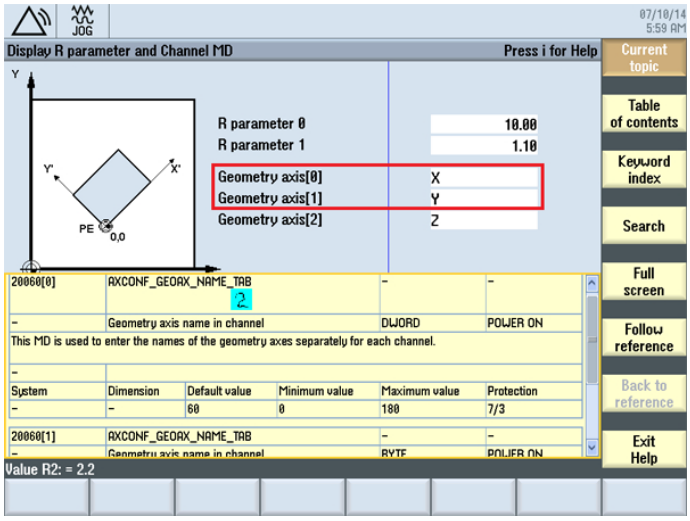



Figure 2-2 Geometry axes with context sensitive help

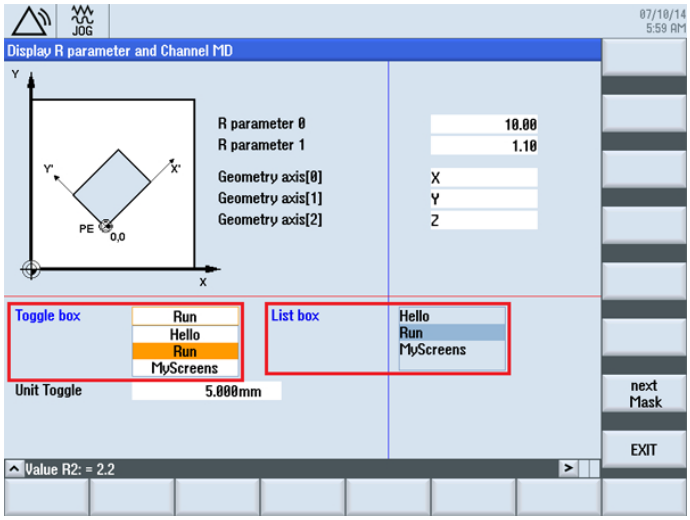



Figure 2-3 Toggle fields: List and box

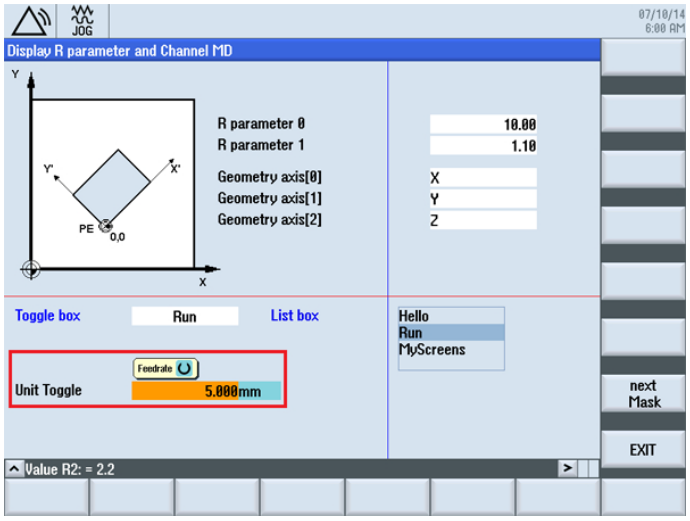



Figure 2-4 I/O field with integrated unit selection

## Dialog 2

MCS and WCS values are displayed in the second dialog. The dialog also contains examples of progress bars with and without color change.

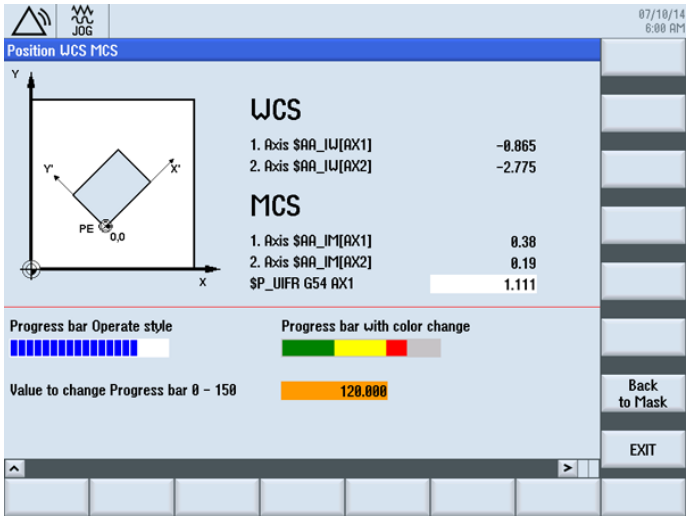



Figure 2-5 Progress bars with (right) and without (left) color change

## Navigation

The first dialog is called using the "START" softkey in the diagnostics operating area. The horizontal SK7 softkey is used.

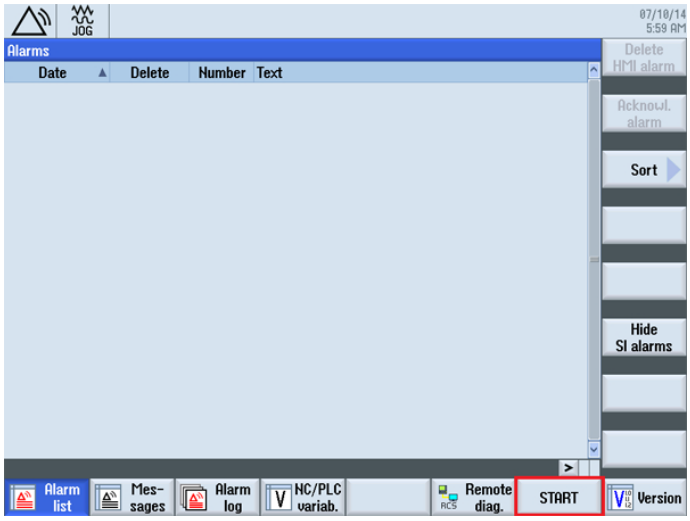



Figure 2-6 Start softkey „START“ in the machine operating area, AUTO mode

You can call the second dialog from the first dialog with the "Next Mask" softkey. With the "EXIT" softkey, you can return to the root screen of the operating area (see diagram above). Using the "EXIT" softkey, you can also return to the root screen of the operating area from the second dialog (see the diagram above). You can return to the first dialog via the "Back to Mask" softkey.

## Procedure

The necessary steps are described in the following chapters:

1. Creating the configuration file (COM file)
2. Saving the configuration file in the OEM directory
3. Creating the online help
4. Saving the online help in the OEM directory
5. Copying easyscreen.ini into the OEM directory
6. Registering the COM file in easyscreen.ini
7. Testing the project

## 2.2.2 Creating the configuration file

### Content of the configuration file

Using a UTF8-capable editor, create the configuration file diag.com for the two dialogs.

```
; Start identifier, start softkey
//S(START)
   ; Start softkey only text
   HS7=("START")
   ; Start softkey with language-dependent text and png
   ;HS7=([$80792,"\\sk_ok.png"])

   ; Press method
   PRESS(HS7)
      ; LM or LS function
      LM("MASK1")
      ; LM, specifying a com file
      LM("MASK1","TEST.COM")
   END_PRESS
; End identifier of start softkey
//END

; Definition of dialog 1 with header and screen
//M(MASK1/"Display R parameter and Channel MD"/"mz961_01.png")
   ; Definition of the variables
   DEF VAR1 = (R2///,"R parameter 0"///"$R[0]"/200,50,150/400,50,100,)
   ; With help screen
   DEF VAR2 = (R2///,"R parameter 1"//"mz961_02.png"/"$R[1]"/200,70,150/400,70,100)
   ; With online help
   DEF ACHS_NAM1 = (S///"Press i for Help","Geometry axis[0]"////200,100,150/400,100,100//"sinumer-
ik_md_1.html","20060[0]")
   ; With online help
   DEF ACHS_NAM2 = (S///"Press i for Help","Geometry axis[1]"////200,120,150/400,120,100//"sinumer-
ik_md_1.html","20060[1]")
   DEF ACHS_NAM3 = (S///,"Geometry axis[2]"////200,140,150/400,140,100)
   ; Definition of toggle fields and unit toggle
   DEF VAR_TGL = (S/* "Hello", "Run", "MyScreens"/"Run"/,"Toggle box"////10,230,100/120,230,100/,,6)

   DEF VAR_TGB = (S/* "Hello", "Run", "MyScreens"/"Run"/,"List box"/
dt4///250,230,100/370,230,100,60/,,"#0602ee")
   ; BC, FC, BC_ST, FC_ST, BC_GT, FC_GT, BC_UT, FC_UT, SC1, SC2
   DEF VarEdit = (R///,"Unit Toggle",,,"Feedrate"///"$R[11]"/5,300,100/120,300,100///"VarTgl"),
VarTgl = (S/*0="mm",1="inch"/0//WR2////220,300,40)
```

```
; Softkey definition in the dialog
HS1=("")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("Next Mask")
VS8=("EXIT")

; Definition LOAD block
LOAD
   ; Read value with RNP
   ACHS_NAM1 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[0]")
   ACHS_NAM2 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[1]")
   ACHS_NAM3 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[2]")
   ; Output of a dialog line
   DLGL("Value R2: = " << RNP("$R[2]"))
   ; Write WNP to a value
   WNP("$R[3]",VAR0)

  ; Vertical dividing line
  V_SEPARATOR(360,1,6,1)
  ; Horizontal dividing line
  H_SEPARATOR(220,1,7,1)
END_LOAD

; Press method
PRESS(VS7)
   ; Load an additional dialog
   LM("MASK2")
; End identifier press method
END_PRESS

; Press method
PRESS(VS8)
```

```
      ; Exit the dialog
      EXIT
   ; End identifier press method
   END_PRESS
; End identifier dialog 1
//END

; Definition of dialog 2 with header and screen
//M(MASK2/"Position WCS MCS"/"mz961_01.png")
   ; Definition of the variables
   DEF TEXT1 = (I///,"WCS"/WR0,fs2///230,30,120/,,1)
   DEF VAR1 = (R3///,"1. Axis $AA_IW[AX1]"/WR1//"$AA_IW[AX1]"/230,70,150/400,70,100)
   DEF VAR2 = (R3///,"2. Axis $AA_IW[AX2]"/WR1//"$AA_IW[AX2]"/230,90,150/400,90,100)

   DEF TEXT2 = (I///,"MCS"/WR0,fs2///230,120,120/,,1)
   DEF VAR3 = (R2///,"1. Axis $AA_IM[AX1]"/WR1//"$AA_IM[AX1]"/230,160,150/400,160,100)
   DEF VAR4 = (R2///,"2. Axis $AA_IM[AX2]"/WR1//"$AA_IM[AX2]"/230,180,150/400,180,100)
   DEF VAR5 = (R3///,"$P_UIFR G54 AX1"///"$P_UIFR[1,AX1,TR]"/230,200,150/400,200,100)

   ; Definition of the progress bar
   DEF PROGGY0 = (R/0,150//,"Progress bar Operate style"/DT2,DO0//"$R[10]"/5,240,190/5,260,150/6,10)
   DEF PROGGY4 = (R/0,150,50,100/120/,"Progress bar with color change"/DT1,DO0//"$R[10]"/
260,240,190/260,260,150/3,4,,,9,7)

   ; Definition of the variables
   DEF PROGVAL = (R/0,150//,"Value to change Progress bar 0 - 150"///"$R[10]"/5,300,230/260,300,100)

  ; Softkey definition in the dialog
  HS1=("")
  HS2=("")
  HS3=("")
  HS4=("")
  HS5=("")
  HS6=("")
  HS7=("")
  HS8=("")
  VS1=("")
  VS2=("")
  VS3=("")
  VS4=("")
  VS5=("")
  VS6=("")
  VS7=("Back to Mask")
  VS8=("EXIT")
```

```
   ; Load method
   LOAD
      H_SEPARATOR(230,1,7,1)
   END_LOAD

   ; Change method
   CHANGE(VAR5)
      ; Function PI_START
      PI_START("/NC,201,_N_SETUFR")
   ; End identifier change method
   END_CHANGE

   ; Press method
   PRESS(RECALL)
      ; Return to the calling mask
      LM("MASK1")
   ; End identifier press method
   END_PRESS

   ; Press method
   PRESS(VS7)
      ; Return to the calling mask
      LM("MASK1")
   ; End identifier press method
   END_PRESS

   ; Press method
   PRESS(VS8)
      ; Exit the dialog to the standard application
      EXIT
   ; End identifier press method
   END_PRESS
; Dialog end identifier
//END
```

## 2.2.3 Saving the configuration file in the OEM directory

### Storage path

Save the configuration file diag.com under the following path:

*[System oem directory]*\proj

## 2.2.4 Creating the online help

### Content of the online help

Create the HTML file sinumerik_md_1.html. The section below shows the context-sensitive help call via **name="20060[0]"** and **name="20060[1]"**. The help file is not commented, as this manual does not discuss creating online help. Notes regarding the integration of online help are provided in Section Configuring the online help (Page 58).

```
<html><head><meta http-equiv="Content-Type" content="text/html; charset="UTF-8"/><title></
title></head>
   <body>
      <table border="1" cellspacing="0" cellpadding="2" width="100%">
         <tr>
            <td><a name="20060[0]"><b>20060[0]</b></a></td>
            <td colspan="3">AXCONF_GEOAX_NAME_TAB</td>
            <center>
               <img src="../picture/bitmap3.png">
            </center>
            <td>-</td>
            <td>-</td>
         </tr>
         <tr>
            <td>-</td>
            <td colspan="3">Geometry axis name in channel</td>
            <td>DWORD</td>
            <td>POWER ON</td>
         </tr>
         <tr>
            <td colspan="6">This MD is used to enter the names of the geometry axes
separately for each channel.<br />
         <tr>
            <td>-</td>
            <td colspan="5"> </td>
         </tr>
         <tr>
          <td width="16*">System</td><td width="16*">Dimension</td><td width="16*">Default
value</td>
            <td width="16*">Minimum value</td><td width="16*">Maximum value</td>
            <td width="16*">Protection</td>
         </tr>
         <tr>
            <td>-</td>
            <td>-</td>
            <td>60</td>
            <td>0</td>
            <td>180</td>
            <td>7/3</td>
         </tr>
      </table>
      <p></p>
      <table border="1" cellspacing="0" cellpadding="2" width="100%">
         <tr>
            <td><a name="20060[1]"><b>20060[1]</b></a></td>
            <td colspan="3">AXCONF_GEOAX_NAME_TAB</td>
            <td>-</td>
            <td>-</td>
         </tr>
         <tr>
            <td>-</td>
            <td colspan="3">Geometry axis name in channel</td><td>BYTE</td>
            <td>POWER ON</td>
         </tr>
         <tr>
```

```
            <td colspan="6">This MD is used to enter the names of the geometry axes
separately for each channel.<br />
         <tr>
            <td>-</td>
            <td colspan="5"> </td>
         </tr>
         <tr>
            <td width="16*">System</td>
            <td width="16*">Dimension</td>
            <td width="16*">Default value</td>
            <td width="16*">Minimum value</td>
            <td width="16*">Maximum value</td>
            <td width="16*">Protection</td>
         </tr>
         <tr>
            <td>-</td>
            <td>-</td>
            <td>0</td>
            <td>0</td>
            <td>2</td>
            <td>7/3</td>
         </tr>
      </table>
      <p></p>
   </body>
</html>
```

## 2.2.5 Saving the online help in the OEM directory

### Storage path

Save the HTML file sinumerik_md_1.html for the English help under the following path:

*[System oem directory]*\hlp/eng

You must create a folder for additional languages (e.g. chs, deu, esp, fra, ita …).

A list of language codes is provided in the Appendix.

## 2.2.6 Copying easyscreen.ini into the OEM directory

### Storage path

Copy the file easyscreen.ini from the directory

*[System SIEMENS directory]*\cfg

to the directory

*[System oem directory]*\cfg

## 2.2.7 Registering the COM file in easyscreen.ini

### Adaptation in easyscreen.ini

Make the following change in the easyscreen.ini in the OEM directory. You have thus registered the diag.com configuration file.

```
;########################
;# AREA Diagnosis       #
;########################
;<============================================================>
;< OEM Softkey on first horizontal Main Menu                   >
;< SOFTKEY position="7"                                        >
;<============================================================>
StartFile28 = area := AreaDiagnosis, dialog:= SlDgDialog, menu := DgGlobalHu, startfile := diag.com
```

## 2.2.8 Testing the project

### Testing the dialog call

Switch to the diagnostics operating area. Click the "START" horizontal softkey.

If "Run MyScreens" detects errors when interpreting the configuration files, these errors are stored in the easyscreen_log.txt text file (see Chapter Troubleshooting (log book) (Page 31)).

### Testing context sensitive help screens and online help

Test the context sensitive help screens and the online help. If errors occur, then check the paths where files have been saved.

# Fundamentals 3

## 3.1 Structure of configuration file

### Introduction

The description of new user interfaces is stored in configuration files. These files are automatically interpreted and the result displayed on the screen. Configuration files are not stored in the software supplied and must be set up by the user.

A UTF-8-capable editor (e.g. the integrated editor of the user interface or Notepad) is used to create the configuration files and language files. The description can also be explained using comments. A ";" is inserted as comment character before every explanation.

**Note**

Make sure that the coding is set to UTF-8 in the editor that you are using when saving the configuration and language files.

However, keywords - also in an UTF-8-coded COM file - may only comprise characters that are contained in the ASCII character set. Otherwise the interpretation and therefore the display of the screen forms/dialogs cannot be guaranteed.

### Configuration

Each HMI operating area has permanent start softkeys which can be used to access newly generated dialogs.

In the event of a "Load a screen form" (LM) or a "Load softkey menu" (LS) call in a configuration file, a new file name containing the object called can be specified. This makes it possible to structure the configuration, e.g. all functions in one operating level in a separate configuration file.

### Structure of the configuration file

A configuration file consists of the following elements:

1. Description of the start softkeys
2. Definition of the dialogs
3. Definition of the variables
4. Description of the methods
5. Definition of the softkey menus

**Note**

**Sequence**

The specified sequence in the configuration file must be maintained.

Example:

```
//S (START)                  ; Definition of the start softkey (optional)
....
//END
//M (.....)                  ; Definition of the dialog
DEF .....                    ; Definition of the variables
LOAD                         ; Description of the blocks
...
END_LOAD
UNLOAD
...
END_UNLOAD
...
//END
//S (...)                    ; Definition of a softkey menu
//END
```

## Storage location for configuration files

The configuration files are located in the *[System user directory]*\proj directory and also correspondingly stored in the *[System add_on directory]* and *[System oem directory]* directories.

## Converting texts from other HMI applications

Procedure to convert a text file with code page coding to text-coding UTF-8:

1. Open the text file on a PG/PC in a text editor.
2. When saving, set the UTF-8 coding.

The read-in mechanism via code page code is still supported.

# 3.2 Structure of the menu tree

## Menu tree principle

Several interlinked dialogs create a menu tree. A link exists if you can switch from one dialog to another. You can use the newly defined horizontal/vertical softkeys in this dialog to call the preceding or any other dialog.

A menu tree can be created behind each start softkey:

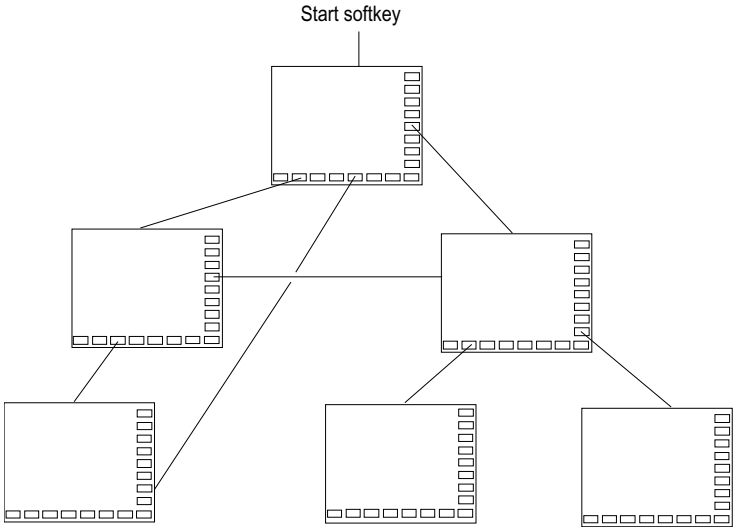



Figure 3-1 Menu tree

## Start softkeys

One or more softkeys (start softkeys), which are used to initiate your own operating sequences, are defined in the configuration files specified in easyscreen.ini.

The loading of a dedicated dialog is associated with a softkey definition or another softkey menu. These are then used to perform the subsequent actions.

Pressing the start softkey loads the assigned dialog. This will also activate the softkeys associated with the dialog. Variables will be output to the standard positions unless specific positions have been configured.

## Reverting to the standard application

You can exit the newly created dialog and return to the standard operating area.

You can use the <RECALL> key to close new user interfaces if you have not configured this key for any other task.

**Note**

**Calling dialogs in the PLC user program**

Dialogs can be selected from the PLC as well as via softkeys: An interface signal is available in DB19.DBB10 for signal exchange between the PLC → HMI.

# 3.3 Definition and functions for start softkeys

## 3.3.1 Defining the start softkey

### Dialog-independent softkey

Start softkeys are dialog-independent softkeys which are not called from a dialog, but which have been configured **before** the first new dialog. In order to access the start dialog or a start softkey menu, the start softkey must be defined.

### Programming

The definition block for a start softkey is structured as follows:

```
//S(Start)                  ; Start identifier of start softkey
HS1=(...)                   ; Define the start softkey: Horizontal SK 1
PRESS(HS1)                  ; Method
  LM...                     ; LM or LS function
END_PRESS                   ; End of method
//END                       ; End identifier of start softkey
```

### Permissible positions for start softkeys

The following positions for "Run MyScreens" start softkeys are permissible in the operating areas:

| Operating area | Position |
|---|---|
| Machine | HSK6 |
| Parameter | HSK7 |
| Program | HSK6 and HSK15<br>Measuring cycles: HSK13 and HSK14 |
| Program manager | HSK2-8 and HSK12-16, if not assigned to drives. |

| Operating area | Position |
|---|---|
| Diagnostics | HSK7 |
| Startup | HSK7 |

Start softkeys are configured in special files. The names of these files are stated in the easyscreen.ini file. They usually have a name which is specific to an operating area (e.g. startup.com for the startup area). The machine operating area represents an exception. There are several mode-specific files in the machine operating area (ma_jog.com, ma_auto.com).

The softkey menu with the start softkeys is called "Start". Existing configurations for start softkeys can still be used. The function whereby start softkeys are merged with the softkeys for the respective HMI application (operating area) in the start softkey menu is not supported.

This means that until the first dialog call is made - in other words, the time at which full functionality becomes available (e.g. execution of PRESS blocks) - menus or softkey menus can only be replaced by others in their entirety.

### Template for configuration of the start softkey

A detailed description of all permissible positions for start softkeys and their configuration is located in the easyscreen.ini file in the following directory:

*[System SIEMENS directory]*\cfg

This file is used as a template for your own configuration of the start softkey.

### See also

Lists of start softkeys (Page 199)

## 3.3.2 Functions for start softkeys

### Functions for dialog-independent softkeys

Only certain functions can be initiated with start softkeys.

The following functions are permitted:

- The **LM function** can be used to load another dialog: **LM***("Identifier"["File"])*
- The **LS function** can be used to display another softkey menu: **LS***("Identifier"[, "File"][, Merge])*
- You can use the **"EXIT" function** to exit newly configured user interfaces and return to the standard application.
- You can use the **"EXITLS" function** to exit the current user interface and load a defined softkey menu.

## PRESS method

The softkey is defined within the definition block and the "LM" or "LS" function is assigned in the PRESS method.

If the start softkey definition is designated as a comment (semicolon (;) at beginning of line) or the configuration file removed, the start softkey will not function.

```
//S(Start)                          ; Start identifier
HS6=("1st screen form")             ; Horizontal SK 6 labeled "1st screen form"
PRESS(HS6)                          ; PRESS method for horizontal SK 6
  LM("Screen form1")                ; Load screen form1 function, where screen form 1
                                    must be defined within the same file.
END_PRESS                           ; End of PRESS method
HS7=("2nd screen form")             ; Horizontal SK 7 labeled "2nd screen form"
PRESS(HS7)                          ; PRESS method for horizontal SK 7
  LM("Screen form2")                ; Load screen form2 function, where screen form2
                                    must be defined within the same file.
END_PRESS                           ; End of PRESS method
//END                               ; End identifier of entry block
```

## Example

```
HS1 = ("new softkey menu")
HS2=("no function")
PRESS(HS1)
  LS("Menu1")                          ;Load new softkey bar
END_PRESS
PRESS (HS2)                            ; Empty PRESS method
END_PRESS
```

## Various configurations of the start softkeys

Various configurations of the start softkeys are merged. In this case, initially the name of the file to be interpreted is read-out of easyscreen.ini. A search is made for files with the .com extension in the following directories:

- *[System user directory]*\proj
- *[System oem directory]*\proj
- *[System add_on directory]*/proj
- *[System Siemens directory]*\proj

The configurations included for the start softkeys are now merged to form a configuration, i.e. the individual softkeys are compared. If there are two or more configurations for a softkey, the higher order configuration is always transferred into the merge version.

Softkey menus or dialogs that are possibly included are ignored. If a softkey has a command without file information, e.g. LM ("test"), as the required softkey menu or dialog is contained in the same file, then the corresponding file name is supplemented in the internal merge version so that in this case, no changes are required. The merge configuration contained is then subsequently displayed.

## 3.4 Troubleshooting (log book)

### Overview

If "Run MyScreens" detects errors when interpreting the configuration files, these errors will be written to the easyscreen_log.txt text file. Only errors of the currently selected dialog are entered. Error entries from previously selected dialogs are deleted.

The file indicates:

- The action during which an error occurred
- The line and column number of the first faulty character
- The entire faulty line of the configuration file
- The entries made by the DEBUG function

**Note**

Each entry has a prefix with the current time stamp in square brackets. This can be helpful, for example, when analyzing time-critical configurations.

### Saving easyscreen-log.txt

The file easyscreen_log.txt is saved in the following directory:

*[System user directory]*\log

### Syntax

The system does not start to interpret the syntax until the start softkey has been defined and a dialog with start and end identifiers as well as a definition line has been configured.

```
//S(Start)
HS6=("1st screen")
PRESS(HS6)
  LM("Maske1")
END_PRESS
//END


//M(Maske1)
```

```
  DEF Var1=(R)
  DEF VAR2 = (R)
  LOAD
  VAR1 = VAR2 + 1              ; Error message in log book, as VAR2 has no value
...
//END

; The right entry would be,
for example:
//M(Maske1)
   DEF Var1=(R)
   DEF VAR2 = (R)

   LOAD
      VAR2 = 7
      VAR1 = VAR2 + 1 ;
...
```

## 3.5 Notes on the easyscreen.ini

As of SINUMERIK Operate V4.7, the easyscreen.ini has been extended by the entries described in this section. The easyscreen.ini can be found in the *[System Siemens directory]*\cfg directory.

### Help display start position

**Entry in easyscreen.ini:**

```
[GENERAL]
HlpPicFixPos=true
```

**Note:**

The start position of help displays is positioned at the configured pixel position independent of the resolution (default=true).

### Elongation behavior, default line height and line spacing

**Entry in easyscreen.ini:**

```
[GENERAL]
SymmetricalAspectRatio=false
DefaultLineHeight=18
```

```
DefaultLineSpacing=3
```

**Notes:**

- The "SymmetricalAspectRatio" entry determines whether the same aspect ratio is used in the X and Y direction when adapting a configuration to a specific screen resolution.
  - "false" (default): Fields and graphics are shortened in widescreen resolution in the Y direction (asymmetrical elongation in relation to 640x480). For example, a square configured in 640x480 is shortened vertically for a widescreen panel and becomes a rectangle.
  - "true": The same aspect ratio is used in the X and Y direction and fields and graphics retain their originally configured proportions in relation to 640x480. For example, a square configured in 640x480 remains a square in a widescreen panel.
- The entries "DefaultLineHeight" and "DefaultLineSpacing" can be used to specify the default line height (default: 18 pixels) and the default line spacing (default: 3 pixels) in relation to 640x480. They only take effect when no Y position or height is specified in the configuration for the position of the short text or the input/output field.

## Resolution-dependent screen position: Form panels

**Entry in easyscreen.ini:**

```
[640x480]
MyPanel = x:=0, y:=220, width:=340, height:=174
[800x480]
MyPanel = x:=0, y:=220, width:=420, height:=174
...
```

**Notes:**

The screen position can be defined as follows:

- The screen position can be specified in "Pixels in relation to 640x480".
  Example:
  `//M(MyMask/"MyCaption"/"\\myhelp.png"/`**0,219,335,174**`)`
- The screen position can be coupled to the resolution-dependent position of a form panel from an Operate standard screen layout, e.g. "Form panel4" ("auxiliary functions") from the screen layout "slmastandardscreenlayout.SlMaStandardScreenLayout" of the "Machine" area
  `//M(MyMask/"MyCaption"/"\`
  `\myhelp.png"/`**"slmastandardscreenlayout.SlMaStandardScreenLayout.FormPanel4"**`)`

  It is therefore easy to superimpose standard forms from Operate or position Easyscreen screens precisely at their position.
  Example:
  `//M(Mask/"Mask"//`**"slstandardscreenlayout.SlStandardScreenLayout.LowerForm"**`)`
  Any other screen layout can also be used.
- The screen position can be coupled to a self-defined, resolution-dependent definition of form panels in the easyscreen.ini file.
  Example:
  `//M(MyMask/"MyCaption"/"\\myhelp.png"/`**"MyPanel"**`)`

## 3.6 Notes for personnel changing over to "Run MyScreens"

**Note**

When using HMI Operate in the NCU, note that all file names are saved in lower case letters on the CF card (com, png, txt).

**Note**

Make sure that the coding is set to UTF-8 in the editor that you are using when saving the configuration and language files.

### Image files

Image files must be saved in the PNG format (xxx.png).

The data must be saved, e.g. for OEM modifications, under

*[System oem directory]*ico/[Resolution]

For further information, see also Chapter Using display images/graphics (Page 50).

### Adapting the configuration file

Check your configuration files regarding the following points:

- Compare the start softkeys with the currently permissible softkeys (see the list of start softkeys in the Appendix) and if required, adapt them.
- Rename the integrated image files corresponding to the "Image files" point above.

The data is stored, e.g. for OEM modifications, in the following directory:

*[System oem directory]*\proj

*[System user directory]*\proj

*[System add_on directory]*\proj

### Adapting the help files

All help files must be saved in UTF-8. Check your existing files, and resave them accordingly with a suitable editor.

The HTML files are saved in the following directory, e.g. for English:

*[System oem directory]*\hlp/eng

*[System user directory]*\hlp/eng

*[System add_on directory]*\hlp/eng

You must create directories for additional languages corresponding to the language identifiers (see the Appendix).

### Checking the "Run MyScreens" license

Check whether the number of dialogs that you have entered exceeds the basic scope of a maximum of five dialogs.

To expand the number of dialogs, you require one of the following software options:

- SINUMERIK 828D/840D sl, SINUMERIK Integrate Run MyScreens (6FC5800-0AP64-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyScreens + Run MyHMI (6FC5800-0AP65-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyHMI / 3GL (6FC5800-0AP60-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyHMI / WinCC (6FC5800-0AP61-0YB0)

# 3.7 Extended configuration syntax

As of SINUMERIK Operate V4.7, a simplified syntax is available for the definition of screens, variables, softkeys and for table columns. Reading and servicing are improved with this alternative syntax. The properties and attributes can be specified in any order, empty entries are omitted. Instead of with round brackets "**(**" and "**)**" as in the previous syntax, the listing of the properties and attributes is now enclosed in curly brackets "**{**" and "**}**".

The properties and attributes are specified as follows:
`{<Name> = <Value>, <Name> = <Value>, …}`

The previous syntax remains compatible.

## Extended syntax for the definition of screens

//M {<Screen name> [,**HD**=<Header>] [,**HLP**=<Graphic>] [,**X**=<X position>] [,**Y**=<Y position>] [,**W**=<Width>] [,**H**=<Height>] [,**VAR**=<>System or user variable] [,**HLP_X**=<X position help display>] [,**HLP_Y**=<Y position help display>] [,**CM**=<Column alignment>] [,**CB**=Behavior when opening the dialog] [,**XG**=<Interpret help display as X3d graphic>] [,**PANEL**=<Name of the linked form panel>][,**MC**=<Screen background color>][,**AL**=<Screen header alignment>]}

**Example:**
```
//M{VariantTest, HD="My Mask"}
```

## Extended syntax for the definition of variables

DEF <Variable name> = {[**TYP**=<Type>] [,**MIN**=<Minimum value>] [,**MAX**=<Maximum value>] [,**TGL**=<Toggle values>] [,**VAL**=<Default>] [,**LT**=<Long text>] [,**ST**=<Short text>] [,**GT**=<Graphic text>] [,**UT**=<Unit text>] [,**TT**=<Tooltip text>] [,**TG**=<Toggle option>] [,**WR**=<Input mode>] [,**AC**=<Access level>] [,**AL**=<Text alignment>] [,**FS**=<Font size>] [,**LI**=<Limit value processing>] [,**UR**=<Update rate>] [,**CB**=<Behavior when opening the dialog>] [,**HLP**=<Help display>] [,**VAR**=<System or user variable>] >] [,**TXT_X**=<X position short text>] [,**TXT_Y**=<Y position short text>] [,**TXT_W**=<Width short text>] [,**TXT_H**=<Height short text>] [,**X**=<X position input/output field>] [,**Y**=<Y position input/output field>] [,**W**=<Width input/output field>] [,**H**=<Height input/output field>] [,**BC**=<Background color input/output field>] [,**FC**=<Foreground color input/output field>] [,**BC_ST**=<Background color short text>] [,**FC_ST**=<Foreground color short text>] [,**BC_GT**=<Background color graphic text>] [,**FC_GT**=<Foreground color graphic text>] [,**BC_UT**=<Background color unit text>] [,**FC_UT**=<Foreground color unit text>] [,**SC1**=<Signal color 1 for progress bar>] [,**SC2**=<Signal color 2 for progress bar>] [,**SVAL1**=<Threshold 1 for progress bar>] [,**SVAL2**=<Threshold 2 for progress bar>] [,**DT**=<Display type>] [,**DO**=<Display alignment>] [,**OHLP**=<Online help>][,**LINK_TGL**=<Name of the linked toggle variable>]}

**Examples:**

```
DEF MyVar5={TYP="R2", ST="MyVar5", VAL=123.4567, OHLP="myhelp.html", MIN=100.1,
MAX=200.9}
DEF MyVar2={TYP="I", TGL="*1,2,3", VAL=1}
DEF MyVar3={TYP="R2", TGL="*0=""Off"", 1=$80000", VAL=1}
DEF MyVar4={TYP="R2", TGL="*MyArray",VAL=1}
DEF MyVar1={TYP="R2", TGL="%grid99", X = 0, W=300, H=200}
```

```
DEF MyVar6={TYP="R2", TGL="+$80000", VAR="$R[10]", ST="Text offset"}
```

## Extended syntax for the definition of softkeys

SK = {[**ST**=<Label>] [,**AC**=<Access level>] [,**SE**=<Status>]}

**Examples:**

```
HS1={ST="""MySk""", AC=6, SE=1}
HS3={ST="SOFTKEY_CANCEL"}
HS5={ST="[$81251,""\\sk_ok.png""]"}
HS8={ST="[""Test"",""\\sk_ok.png""]"}
```

## Extended syntax for the definition of table columns

{[**TYP**=<Type>] [,**MIN**=<Minimum value>] [,**MAX**=<Maximum value>] [,**LT**=<Long text>] [,**ST**=<Short text>] [,**WR**=<Input mode>] [,**AC**=<Access level>] [,**AL**=<Text alignment>] [,**FS**=<Font size>] [,**LI**=<Limit value processing>] [,**UR**=<Update rate>] [,**HLP**=<Help display>] [,**VAR**=<System or user variable>] >] [,**W**=<Column width>] [,**OF1**=<Offset1>] [,**OF2**=<Offset2>] [,**OF3**=<Offset3>]}

**Example:**

```
DEF MyGridVar={TYP="R", TGL="%MyGrid1", X=10, W=550, H=100}
//G(MyGrid1/0/5)
   {TYP="I", ST="Index", WR=1, VAR="1", W=80, OF1=1}
   {TYP="S", LT="LongText2", ST="Text", WR=1, VAR="$80000", AL=2, W=330, OF1=1}
   {TYP="R3", LT="LongText1", ST="R9,R11,R13,R15", WR=2, VAR="$R[1]", W=110, OF1=2}
//END
```

# Dialogs

# 4

## 4.1 Structure and elements of a dialog

### 4.1.1 Defining a dialog

#### Definition

A dialog is part of a user interface consisting of a display line, dialog elements and/or graphics, an output line for messages and 8 horizontal and 8 vertical softkeys.

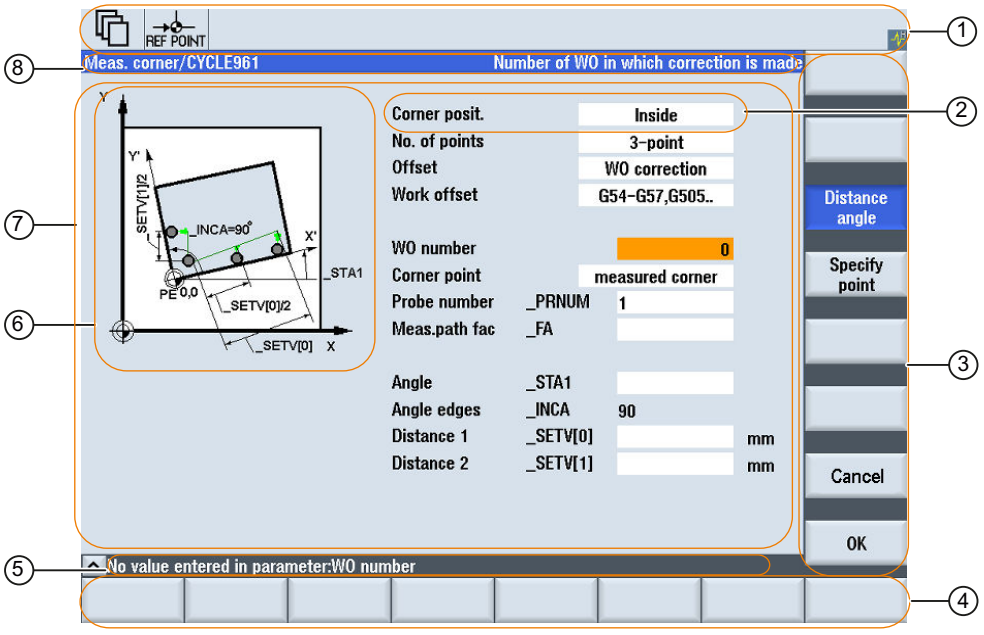
Dialog elements are:

- Variables
  - Limits/toggle field
  - Default setting of variables
- Help display
- Texts
- Attributes
- System or user variable
- Position of short text
- Position of input/output field
- Colors

Dialog properties:

- Header
- Graphic
- Dimension
- System or user variable

- Graphic position
- Attributes




① Machine status display ("header")
② Dialog element
③ 8 vertical softkeys
④ 8 horizontal softkeys
⑤ Output of diagnostic messages
⑥ Graphic
⑦ Dialog
⑧ Header line of the dialog with header and long text

Figure 4-1 Structure of the dialog

## Overview

The definition of a dialog (definition block) is basically structured as follows:

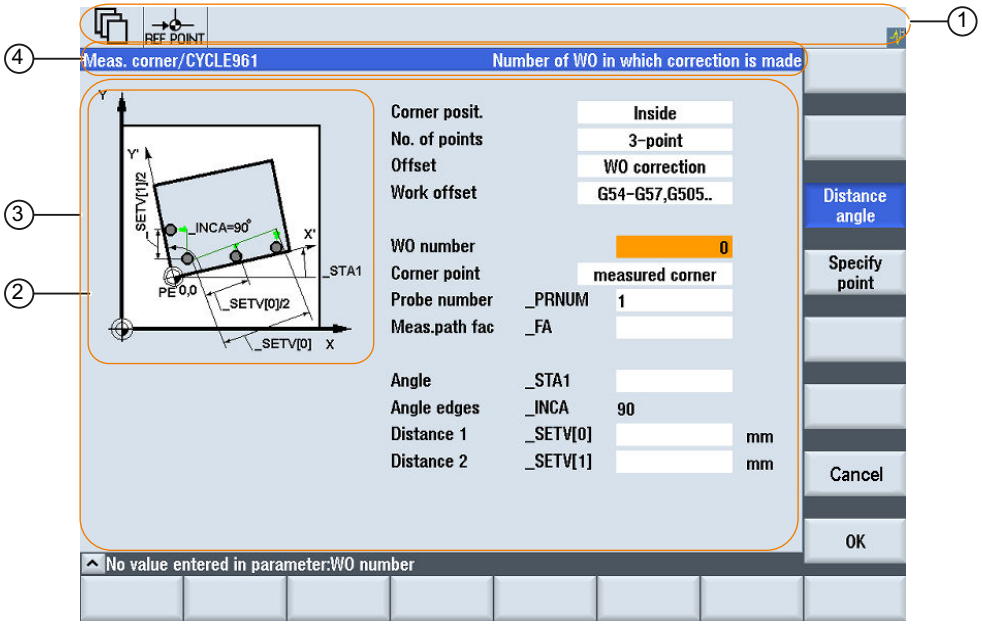
| Definition block | Comment | Chapter reference |
|---|---|---|
| `//M...` | ;Dialog start identifier | |
| `DEF Var1=...`<br>`...` | ;Variables | See chapter "Variables" |
| `HS1=(...)`<br>`...` | ;Softkeys | See chapter "Softkey menus" |
| `PRESS(HS1)`<br>`  LM...`<br>`END_PRESS` | ;Method start identifier<br>;Actions<br>;Method end identifier | See chapter "Methods" |
| `//END` | ;Dialog end identifier | |

Within the dialog definition block, various variables that appear as dialog elements in the dialog, as well as horizontal and vertical softkeys, are defined first. Different types of actions are then configured in methods.

## 4.1.2 Defining dialog properties

### Description

The properties of the dialog are defined in the start identifier line of the dialog.




① Machine status display ("header")
② Graphic
③ Dialog
④ Header line of the dialog with header and long text

Figure 4-2 Dialog properties

### Programming

| | |
|---|---|
| Syntax: | **//M**(Identifier/[Header]/[Graphic]/[Dimension]/[System or user variable]/[Graphic position]/[Attributes])<br>See also Section Extended configuration syntax (Page 36). |
| Description: | Defines a dialog |

| | | | | |
|---|---|---|---|---|
| Parameter: | Identifier | | | Name of the dialog. |
| | Header | | | Dialog header as text or call for text (e.g. $85011) from a language-specific text file. |
| | Graphic | | | Graphics file with path in double quotation marks. |
| | Dimension | | | Position and size of the dialog in pixels (distance from left-hand side, distance from right-hand side, width, height), in relation to the upper left-hand corner of the screen. The entries are separated by a comma. |
| | System or user variable | | | System or user variable to which the current cursor position is assigned. The NC or PLC can be provided with the cursor position via the system or user variable. The first variable has the index 1. The order corresponds to the configuration order of the variables. |
| | Graphic position | | | Position of the graphic in pixels (distance from left-hand side, distance from right-hand side), in relation to the upper left-hand corner of the dialog. The entries are separated by a comma. |
| | Attributes | | | The specifications of the attributes are separated by a comma.<br>Possible attributes are: |
| | | CM | | Column mode: Column alignment |
| | | | CM0 | Default setting: The column distribution is carried out separately for each line. |
| | | | CM1 | The column distribution of the line with the most columns applies to all lines. |
| | | CB | | CHANGE block: Response when dialog is opened: cb attributes specified for a variable in a variables definition take priority over the default setting in the dialog definition.<br>See also Behavior when opening the dialog (attribute CB) (Page 204). |
| | | | CB0 | Default setting: All CHANGE blocks associated with the dialog are processed when it is opened. |
| | | | CB1 | CHANGE blocks are then only processed if the relevant value changes. |
| | | XG | | Integration of an X3D animation as help display (only in the machining step programming).<br>Example:<br>`//M(Meas/ $85605/"myx3dhelpfile.hmi,,Z_Animation,,G17"///30,10/ XG1)` |
| | | | XG0 | Default setting = 0 |
| | | | XG1 | The screen attribute XG must already be set to 1 in the screen definition, a change during runtime is not possible.<br>Note:<br>The specification in the screen property HLP must also be set accordingly. |
| | | AL | | The alignment of the screen header can be influenced with the screen attribute AL.<br>Example:<br>Central setting of the "Set password" window header:<br>`//M(MY_PWD_SET/”Set password”//"EasyPwdModalLayout"// AL2)` |
| | | | AL0 | Left-justified, default |
| | | | AL1 | Right-justified |
| | | | AL2 | Centered |
| | | MC | | Screen background color (screen color) |

| | | | |
|---|---|---|---|
| | | | Example:<br>Set screen background color blue (= 6)<br>`//M(MyMask/"MyMask"//////6)`<br>or<br>`PRESS(HS1)`<br>`  MC=6`<br>`END_PRESS` |

## Accessing the dialog properties

Read and write access to the following dialog properties is permitted within methods (e.g. PRESS block).

- HD = Header
- HLP = Help display
- VAR = System or user variable
- MC = Screen background color
- CM = Column alignment (read-only)
- CB = Behavior when opening (read-only)
- XG = Integration of X3d (read-only)
- AL = Screen header alignment (read-only)

## Example

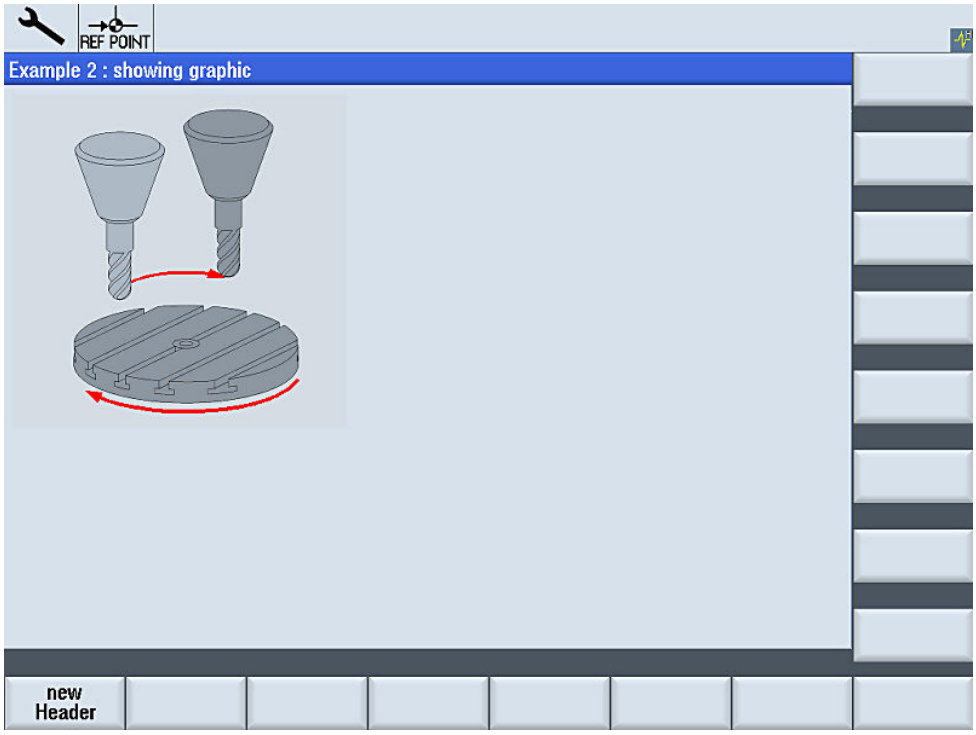



Figure 4-3 "Example 2: showing graphic"

```
//S(Start)
HS7=("Example", se1, ac7)

PRESS(HS7)
  LM("Mask2")
END_PRESS

//END
//M(Mask2/"Example 2 : showing graphic"/"example.png")
HS1=("new%nHeader")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
```

```
VS4=("")
VS5=("")
VS6=("")
VS7=("")
VS8=("")

PRESS(HS1)
  Hd= "new Header"
END_PRESS
...
//END
```

### See also

Programming example for the "Custom" area (Page 188)

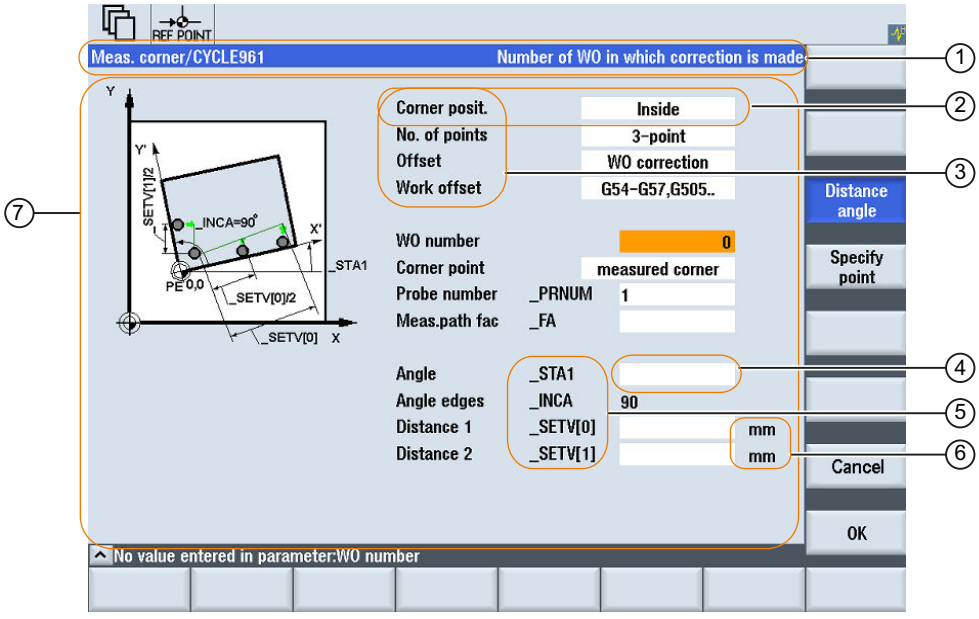
## 4.1.3 Defining dialog elements

### Dialog element

The term "dialog element" refers to the visible part of a variable, i.e. short text, graphics text, input/output field, unit text and tooltip. Dialog elements fill lines in the main body of the dialog. One or more dialog elements can be defined for each line.

## Variable properties

All variables are valid in the active dialog only. Properties are assigned to a variable when it is defined. The values of dialog properties can be accessed within methods (e.g. a PRESS method).




① Header line of the dialog with header and long text
② Dialog element
③ Short text
④ Input/output field
⑤ Graphic text
⑥ Text for units
⑦ Main body of the dialog

Figure 4-4 Elements of a dialog

## Programming - Overview

The single parameters to be separated by commas are enclosed in round brackets:

| DEF *Identifier* = | **Identifier = Name of variable** |
|---|---|
| | Variable type |
| | /[Limit values or toggle field] |
| | /[Default] |
| | /[Texts (Long text, Short text\|Image, Graphic text, Units text)] |
| | /[Attributes] |
| | /[Help display] |
| | /[System or user variable] |
| | /[Position of short text] |
| | /[Position of I/O field(Left, Top, Width, Height)] |

| | |
|---|---|
| | /[Colors] |
| | /[online help] (Page 58) |

### See also

Variable parameters (Page 69)

## 4.1.4 Defining dialogs with multiple columns

### Overview

Multiple variables can also be represented in a dialog on one line. In this case, the variables are all defined in the configuration file on a single definition line.

```
DEF VAR11 = (S///"Var11"), VAR12 = (I///"Var12")
```

To make individual variables in the configuration file more legible, the definition lines can be wrapped after every variables definition and following comma.

The key word "DEF" always indicates the beginning of a new line:

```
DEF Tnr1=(I//1/"","T ",""/wr1///,,10/20,,50),
  TOP1=(I///,"Type="/WR2//"$TC_DP1[1,1]"/80,,30/120,,50),
  TOP2=(R3///,"L1="/WR2//"$TC_DP3[1,1]"/170,,30/210,,70),
   TOP3=(R3///,"L2="/WR2//"$TC_DP4[1,1]"/280,,30/320,,70),
   TOP4=(R3///,"L3="/WR2//"$TC_DP5[1,1]"/390,,30/420,,70)
DEF Tnr2=(I//2/"","T ",""/wr1///,,10/20,,50),
   TOP21=(I///,"Typ="/WR2//"$TC_DP1[2,1]"/80,,30/120,,50),
   TOP22=(R3///,"L1="/WR2//"$TC_DP3[2,1]"/170,,30/210,,70),
   TOP23=(R3///,"L2="/WR2//"$TC_DP4[2,1]"/280,,30/320,,70),
   TOP24=(R3///,"L3="/WR2//"$TC_DP5[2,1]"/390,,30/420,,70)
```

**Note**

When configuring multi-column dialogs, please observe that a large number of columns might slow down the system!

## 4.1.5 Password dialogs

### Using the standard password dialogs

You can integrate the following predefined password dialogs in the screen configuration:

- Set password

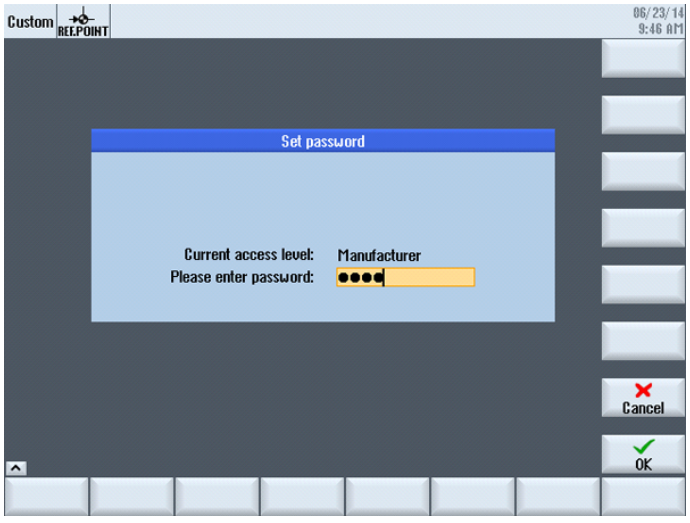



Figure 4-5 Set password dialog

- Change password

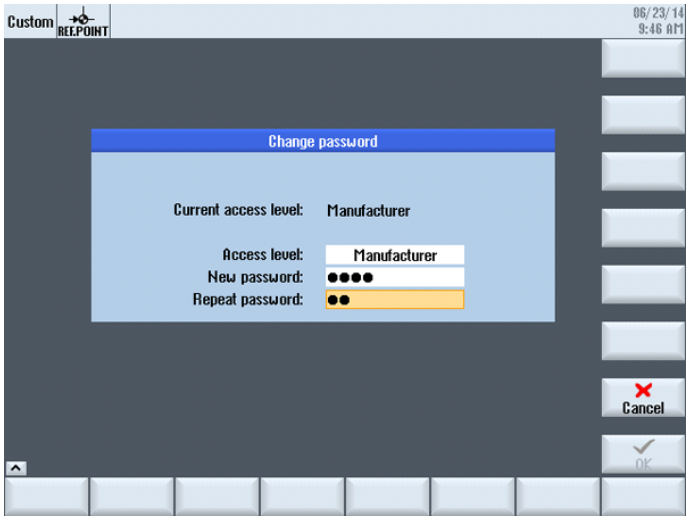



Figure 4-6 Change password dialog

- Delete password

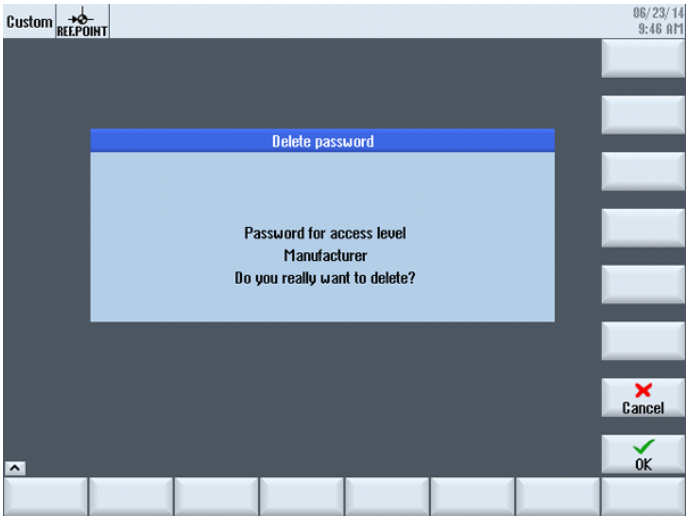



Figure 4-7 Delete password dialog

**Note**

The password functionality is already set with these dialogs. The dialogs do not correspond to the dialogs of SINUMERIK Operate.

The dialogs can be called either via the Load softkey (LS) function or the Load mask (LM) function:

1. Call via the Load softkey (LS) function:
   `LS("Passwd","slespasswd.com",1)`
   Extension of the vertical softkeys:
   - Softkey 1: Set password
   - Softkey 2: Change password
   - Softkey 3: Delete password
2. Alternatively, the three screens can be opened directly by calling the Load mask (LM) function:
   - Set password: `LM("PWD_SET", "slespasswd.com",1)`
   - Change password: `LM("PWD_CHG", "slespasswd.com",1)`
   - Delete password: `LM("PWD_CLEAR", "slespasswd.com",1)`

### 4.1.6 Using display images/graphics

#### Use of graphics

A distinction is made between:

- Display images / graphics in the graphic area.
- Help displays illustrating, for example, individual variables, which are superimposed in the graphic area.
- More Help displays can be configured instead of short text or an input/output field, which you position where you like.

#### Storage locations

The image matching the resolution of the connected monitor is searched for in the associated resolution directories in the following sequence:

*[System user directory]*/ico/ico<Resolution>
*[System oem directory]*/ico/ico<Resolution>
*[System add_on directory]*/ico/ico<Resolution>

If the image is not displayed or not found, copy it into one of the following directories for a resolution of 640 x 480 pixels:

*[System user directory]*/ico/ico640
*[System oem directory]*/ico/ico640
*[System add_on directory]*/ico/ico640

**Note**

With the different panel resolutions, the images are positioned proportionally.

## 4.2 Defining softkey menus

#### Definition

The term softkey menu is used to refer to all the horizontal and vertical softkeys displayed on a screen form. In addition to the existing softkey menus, it is possible to define other menus, which partially or completely overwrite the existing menus.

The names of the softkeys are predefined. Not all softkeys need to be assigned.

HSx x 1 - 8, Horizontal softkeys 1 to 8

VSy y 1 - 8, Vertical softkeys 1 to 8

The definition of a softkey menu (softkey menu definition block) is basically structured as follows:

| Definition block | Comment | Section reference |
|---|---|---|
| `//S...` | ;Start identifier of softkey menu | |
| `HSx=...` | ;Define softkeys | |
| `PRESS(HSx)`<br>`  LM...`<br>`END_PRESS` | ;Method start identifier<br>  ;Actions<br>;Method end identifier | See Section "Methods" |
| `//END` | ;End identifier of softkey menu | |

## Description

Properties are assigned to softkeys during definition of the softkey menu.

## Programming

| | | |
|---|---|---|
| Syntax: | ***//S****(Identifier)*<br>...<br>//END | ;Start identifier of softkey menu<br><br>;End identifier of softkey menu |
| | See also Section Extended configuration syntax (Page 36). | |
| Description: | Defines the softkey menu | |
| Parameter: | Identifier | Name of the softkey menu |
| | Text or image file name | |
| Syntax: | **SK** = *(Text[, Access level][, Status][, Alignment of the softkey image][, Text alignment in relation to the softkey image])* | |
| Description: | Define softkey | |
| Parameter: | SK | Softkey, e.g. HS1 to HS8, VS1 to VS8 |

| | | | |
|---|---|---|---|
| | Text | Enter text | |
| | Display file name | "\\my_pic.png"<br>or via separate text file $85199, e.g. with the following text in the (language-specific) text file: 85100 0 0 "\ \my_pic.png".<br>The image size which can be displayed on a softkey depends on the OP used: | |
| | | OP 08: | 640 x 480 mm → 25 x 25 pixels |
| | | OP 010: | 640 x 480 mm → 25 x 25 pixels |
| | | OP 012: | 800 X 600 mm → 30 x 30 pixels |
| | | OP 015: | 1024 X 768 mm → 40 x 40 pixels |
| | | OP 019: | 1280 x 1024 mm → 72 x 72 pixels |
| | Access level | ac0 to ac7 (ac7: default) | |
| | Status | se1: visible (default)<br>se2: disabled (gray text)<br>se3: displayed (last softkey used) | |
| | Alignment of the softkey image | PA (PictureAlignment) | |
| | | Valid values:<br>0: Left<br>1: Right<br>2: Centered<br>3: Top (default)<br>4: Bottom | |
| | Text alignment in relation to the softkey image | TP (TextAlignedToPicture) | |
| | | Valid values:<br>0: Text is not aligned to the image<br>1: Text is aligned to the image (default) | |

**Note**

Enter %n in the softkey text to create a line break.

A maximum of 2 lines with 9 characters each are available.

## Assigning access level

Operators can only access information on this access level and lower access levels (see also List of access levels (Page 202)).

## Example

```
//S(Menu1)                                ; Start identifier of softkey menu
HS1=("NEW", ac6, se2)                     ; Define softkey HS1, assign the label "NEW", protec-
                                          tion level 6, and the status "disabled"
```

```
HS2=("\\image1.png")                          ; Assign a softkey to the graphic
HS3=("Exit")
HS4=(["Confirm","\\sk_ok.png"],PA0,TP1)       ; Softkey with text and graphic, Text="Confirm", Im-
                                              age="sk_ok.png", Alignment of the softkey image: Left,
                                              text is aligned to the image


VS1=("sub screen form")
VS2=($85011, ac7, se2)                        ; Define softkey VS2, assign the text from the lan-
                                              guage file, protection level 1, and the status "disa-
                                              bled"
VS3=("Cancel", ac1, se3)                      ; Define softkey VS3, assign the label "Cancel", pro-
                                              tection level 1 and the status "highlighted"
VS4=("OK", ac6, se1)                          ; Define softkey VS4, assign the label "OK", protec-
                                              tion level 6 and the status "visible"
VS5=(SOFTKEY_CANCEL,,se1)                     ; Define cancel standard softkey VS5 and assign the
                                              status "visible"
VS6=(SOFTKEY_OK,,se1)                         ; Define OK standard softkey VS6 and assign the status
                                              "visible"
VS7=(["\\image1.png","OEM text"],,se1)        ; Define softkey VS7, assign an image, assign the la-
                                              bel "OEM Text" and the status "visible"
VS8=(["\\image1.png",$83533],,se1)            ; Define softkey VS8, assign an image, assign text
                                              from language file and the status "visible"


PRESS(HS1)                                    ; Method start identifier
  HS1.st="Calculate"                          ; Assign a label text to the softkey
...
END_PRESS                                     ; Method end identifier

PRESS(RECALL)                                 ; Method start identifier
  LM("Screen form21")                         ; Load dialog
END_PRESS                                     ; Method end identifier
//END                                         ; End identifier of softkey menu
```

## 4.2.1 Changing softkey properties during runtime

### Description

The softkey properties Text, Access Level and Status can be changed in the methods during runtime.

## Programming

| | | | |
|---|---|---|---|
| Syntax: | SK.st = "Text"<br>SK.ac = Access level<br>SK.se = Status<br>SK.pa = "Alignment of the softkey image"<br>SK.tp = "Text alignment in relation to the softkey image" | | ;Softkey with label<br>;Softkey with protection level<br>; Softkey with status<br>;Softkey with image<br>;Softkey with image and label |
| Description: | Assign properties | | |
| Parameter: | Text | | Label text in inverted commas |
| | Access level | | Range of values: 0 ... 7 |
| | Status | 1:<br>2:<br>3: | Visible and operator-controllable<br>Disabled (gray text)<br>Highlighted (last softkey used) |
| | Alignment of the softkey image | 0:<br>1:<br>2:<br>3:<br>4: | Left<br>Right<br>Centered<br>Top (default)<br>Bottom |
| | Text alignment in relation to the softkey image | 0:<br>1: | Text is not aligned to the image<br>Text is aligned to the image (default) |

## Example

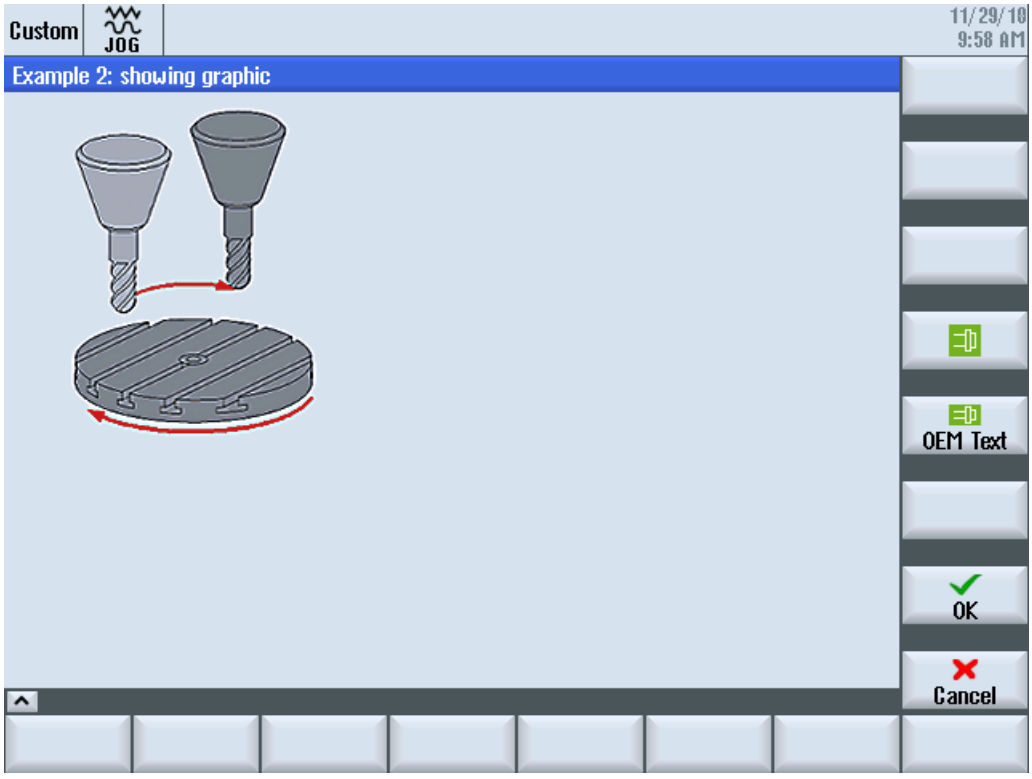



Figure 4-8 Example 3: Graphics and softkeys

```
//S(Start)
HS7=("Example", ac7, se1)

PRESS(HS7)
  LM("Maske3")
END_PRESS

//END

//M(Maske3/"Example 2: showing graphic"/"example.png")
HS1=("")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
```

```
VS1=("")
VS2=("")
VS3=("")
VS4=("\\sp_ok.png",,SE1)
VS5=(["\\sp_ok_small.png","OEM Text"],,SE1)
VS6=("")
VS7=(SOFTKEY_OK,,SE1)
VS8=(SOFTKEY_CANCEL,,SE1)
PRESS(VS4)
  EXIT
END_PRESS
PRESS(VS5)
  EXIT
END_PRESS
PRESS(VS7)
  EXIT
END_PRESS
PRESS(VS8)
  EXIT
END_PRESS
//END
```

## 4.2.2 Language-dependent text

### Overview

Language-dependent texts are used for:

- Softkey labels
- Headings
- Help texts
- Any other texts

The language-dependent texts for dialogs are stored in text files.

The text files are stored in the following directories:

- *[System user directory]*\lng
- *[System oem directory]*\lng
- *[System add_on directory]*\lng

**Note**

The text files must be saved in the same way as the project files.

For example:

*[System user directory]*/lng/[Text file]

*[System user directory]*/proj/[Configuration file]

| | |
|---|---|
| alsc.txt | Language-dependent texts for the Siemens standard cycles |
| almc.txt | Language-dependent texts for the Siemens measuring cycles |

The text files used during program runtime are specified in the easyscreen.ini file:

```
[LANGUAGEFILES]
LngFile03 = user.txt      ;->user<_xxx>.txt (e.g. user_eng.txt)
```

In this instance, the user.txt file has been chosen as an example of a text file. The name can always be freely selected. Depending on the language of the texts within the file, the relevant language code must be added using the following syntax. An underscore followed by the relevant language identifier is added after the name e.g. `user_eng.txt`.

**Note**

You must not use LngFile01 and LngFile02 for user texts because they are already assigned in the standard easyscreen.ini.

## See also

List of language codes used in file names (Page 203)

## Format of text files

The text files must be saved in UTF-8 format.

**Note**

Make sure that the coding is set to UTF-8 in the editor that you are using when saving the configuration and language files.

## Format of a text entry

| | |
|---|---|
| Syntax: | 8xxxx 0 0 "Text" |
| Description: | Assignment between text number and text in the file |

| | | | |
|---|---|---|---|
| Parameter: | xxxx | 85,000 to 89,999 | Text identification number range reserved for users. You must assign unique numbers. |
| | "Text" | | Text that appears in the dialog |
| | %n | | Control characters in the text for creating a line break |

Parameters 2 and 3 are separated by blanks and act as control characters for alarm text output. To ensure that the text format is identical to that of the alarm texts, you must set zero.

**Examples of language-dependent texts:**

```
85000 0 0 "Retraction plane"
85001 0 0 "Drilling depth"
85002 0 0 "Pitch"
85003 0 0 "Pocket radius"
```

## 4.3 Configuring the online help

### Online help

An online help for the configured dialogs and elements can be created in the HTML format.

The syntax and procedure for the online help is essentially the same as for SINUMERIK Operate.

If an online help is configured for an input field, then the standard screens of the online help are used to display the user-specific online help:

```
DEF RFP=(R//1/,"RFP","RFP"///////"sinumerik_md_1.html","9006")
```

**Note**

As a result of LINUX, the HTML file must be written in lower case letters!

The HTML file is saved in the following directory, e.g. for English.

*[System user directory]*/hlp/eng

*[System oem directory]*/hlp/eng

*[System add_on directory]*/hlp/eng

You must create directories for additional languages corresponding to the language identifiers (see the Appendix).

## References

Commissioning Manual "Base software and operating software" (IM9), Chapter "OEM-specific online help"

# Variables

# 5

## 5.1 Defining variables

### Variable value

The major property of a variable is its value.

The value of variables can be assigned by means of:

- Default settings when defining variables
- Assignment to a system or user variable
- A method

### Programming

| | | |
|---|---|---|
| Syntax: | Identifier.**val** = Variable value<br>Identifier = Variable value | |
| Description: | Variable value val (value) | |
| Parameter: | Identifier: | Name of the variable |
| | Variable value: | Value of the variable |
| Example: | `VAR3 = VAR4 + SIN(VAR5)`<br>`VAR3.VAL = VAR4 + SIN(VAR5)` | |

### Variable status

The "Variable status" property scans a variable for valid content during runtime. This property can be read and written with the value FALSE = 0.

### Programming

| | | |
|---|---|---|
| Syntax: | Identifier.**vld** | |
| Description: | Variable status vld (validation) | |
| Parameter: | Identifier: | Name of the variable |
| | | The result of the scan can be: |
| | FALSE = | invalid value |
| | TRUE = | valid value |
| Example: | `IF VAR1.VLD == FALSE`<br>`  VAR1 = 84`<br>`ENDIF` | |

# 5.2 Application examples

## Help variables

Help variables are internal arithmetic variables. Arithmetic variables are defined like other variables, but have no other properties apart from variable value and status, i.e. Help variables are not visible in the dialog. Help variables are of the VARIANT type.

## Programming

| | | |
|---|---|---|
| Syntax: | DEF *Identifier* | |
| Description: | Internal arithmetic variables of the VARIANT type | |
| Parameter: | Identifier: | Name of the Help variable |
| | | |
| Example: | `DEF OTTO ;Definition of a Help variable` | |

| | | |
|---|---|---|
| Syntax: | Identifier.val = *Help variable value*<br>Identifier = *Help variable value* | |
| Description: | A value is assigned to a Help variable in a method. | |
| Parameter: | Identifier: | Name of the Help variable |
| | Help variable value: | Content of the Help variables |

**Example:**

```
LOAD
  OTTO = "Test"             ; The value "Test"  is assigned to help variable Otto.
END_LOAD
LOAD
  OTTO = REG[9].VAL         ; The value of the register is assigned to help variable
END_LOAD                    Otto
```

## Calculation with variables

Variables are calculated every time you exit an IO field (by pressing the ENTER or TOGGLE key). The calculation is configured in a CHANGE method that is processed every time the value changes.

You can scan the variable status to ascertain the validity of the value of the variable, e.g.

```
IF VAR1.VLD == FALSE
   VAR1 = 84
ENDIF
```

### Addressing system variables indirectly

A system variable can also be addressed indirectly, i.e. as a function of another variable:

```
PRESS(HS1)
  ACHSE=ACHSE+1
  WEG.VAR="$AA_DTBW["<<ACHSE<<"]"       ;Address axis address via variable
END_PRESS
```

### Changing softkey labels

Example:

```
HS3.st = "New text"                     ;Change softkey label
```

## 5.3 Example 1: Assigning the variable type, texts, help display, colors, tooltips

### Example 1a

The following example defines a variable for which the properties variable type, texts, help display and colors are set.

| DEF Var1 = (R///,"Actual value",,"mm"//"Var1.png"////8,2) | | | |
|---|---|---|---|
| | Variable type: | | REAL |
| | Texts: | | |
| | | Short text: | Actual value |
| | | Unit text: | mm |
| | Help screen: | | Var1.png |
| | Colors: | | |
| | | Foreground color: | 8 (brown) |
| | | Background color: | 2 (orange) |

### Example 1b

The following example defines a variable for which the properties variable type, default setting, texts, tooltip, input mode and position of short text are set.

| DEF Var2 = (I//5/"","Value","","","" Tooltip text"/wr2///20,250,50) | | |
|---|---|---|
| Variable type: | | INTEGER |
| Default setting: | | 5 |
| Texts: | | |
| | Short text: | Value (possible language text ID) |
| | Tooltip: | ToolTipText |
| Attributes: | | |
| | Input mode: | Reading and writing |
| Position of short text: | | |
| | Distance from left: | 20 |
| | Distance from top: | 250 |
| | Width: | 50 |

### See also

Variable parameters (Page 69)

## 5.4 Example 2: Assigning the Variable Type, Limits, Attributes, Short Text Position properties

### Example 2

The following example defines a variable for which the properties variable type, limits, input mode, alignment and position are set.

| DEF Var2 = (I/0,10///wr1,al1///,,300) | | |
|---|---|---|
| Variable type: | | INTEGER |
| Limits or toggle field entries: | | MIN: 0<br>MAX: 10 |
| Attributes: | | |
| | Input mode: | Read-only |
| | Alignment of short text: | Right-justified |
| Position of short text: | | |
| | Width: | 300 |

### See also

Variable parameters (Page 69)

## 5.5 Example 3: Assigning the Variable Type, Default, System or User Variable, Input/Output Field Position properties

### Example 3

The following example defines a variable for which the properties variable type, default setting, system or user variable and position are set.

| DEF Var3 = (R//10///"$R[1]"//300,10,200//) | | | |
|---|---|---|---|
| | Variable type: | | REAL |
| | Default setting: | | 10 |
| | System or user variable: | | $R[1] (R-Parameter 1) |
| | Position of short text: | | Default position in relation to input/output field |
| | Position of input/output field: | | |
| | | Distance from left: | 300 |
| | | Distance from top: | 10 |
| | | Width: | 200 |

### See also

Variable parameters (Page 69)

## 5.6 Example 4: Toggle field and list field

### Example 4a

Various entries in the toggle field:

```
DEF Var1 = (I/* 0,1,2,3)                ;Simple toggle field to toggle numeric values
DEF Var2 = (S/* "On", "Off")            ;Simple toggle field to toggle strings
DEF Var3 = (B/* 1="On", 0="Off")        ;Extended toggle field to toggle numeric values whereby a
                                         display text is assigned to each numeric value
DEF Var4 = (R/* ARR1)                   ;Toggle field which obtains its values to be toggled from an
                                         array
```

### Example 4b

The list field corresponds to the configuration of a toggle field, but the display type for the list field (variable attribute DT = 4) must also be set.

| DEF VAR_LISTBOX_Text = (S/*$80000,$80001,$80002,$80003,$80004/$80001//DT4////200,,340,60) | | | |
|---|---|---|---|
| | Variable type: | | STRING |
| | Limits / toggle field: | | *$80000,$80001,$80002,$80003,$80004<br>(List of the language-dependent texts to be displayed) |
| | Default setting: | | $80001 |
| | Attributes: | | |
| | | Display type: | 4 (list field) |
| | Position of input/output field: | | |
| | | Distance from left: | 200 |
| | | Width: | 340 |
| | | Height: | 60 |
| | Colors: | | |
| | | Foreground color: | 6 (blue) |
| | | Background color: | 10 (white) |

## 5.7 Example 5: Image display

### Example 5

Displaying an image instead of a short text: The size and position of the image is defined under "Position of input/output field (left, top, width, height)".

| DEF VAR6= (V///,"\\image1.png" ////160,40,50,50) | | | |
|---|---|---|---|
| | Variable type: | | VARIANT |
| | Texts: | | |
| | | Short text: | image1.png |
| | Position of input/output field: | | |
| | | Distance from left: | 160 |
| | | Distance from the top: | 40 |
| | | Width: | 50 |
| | | Height: | 50 |

## 5.8 Example 6: Progress bar

The progress bar is a special display type of the input/output field and is designed for display without input.

Basically there are two progress bar types:

1. Progress bars with up to two color changes, e.g. for temperature or utilization display (see example 6a)
2. Progress bars to display the progress (without color change) in the Operate style (see example 6b)

## Example 6a

Progress bar with two color changes:



Figure 5-1 Progress bar with two color changes

| DEF PROGGY0 = (R/0,150,50,100///DT1,DO0//"$R[10]"//,,150/3,4,,,,,,,9,7) | | |
|---|---|---|
| Variable type: | | REAL |
| Limits / toggle field: | | |
| | MIN: | 0 |
| | MAX: | 150 |
| | Signal value SVAL1: | 50 |
| | Signal value SVAL2: | 100 |
| Attributes: | | |
| | Display mode DT: | 1 (progress bar) |
| | Display option DO: | 0 (from left to right (default)) |
| System or user variable: | | $R[10] |
| Position of input/output field: | | |
| | Width: | 150 |
| Colors: | | |
| | Foreground color: | 3 (dark green) |
| | Background color: | 4 (light gray) |
| | Signal color SC1: | 9 (yellow) |
| | Signal color SC2: | 7 (red) |

To use a progress bar with color change, the display mode DT (DisplayType) must be set to 1.

The orientation of the progress bar is determined via the attribute display option DO (DisplayOption):

0: From left to right (default)

1: From right to left

2: From bottom to top

3: From top to bottom

A MIN and a MAX value must be specified for the display of the progress bar (in the example MIN: 0, MAX: 150).

Depending on the current value of the variable PROGGY0, this setting already displays a progress bar with the foreground color 3 (= dark green) and the background color 4 (= light gray).

Optionally, one or two signal values SVAL1 and SVAL2 (limit parameters) can be defined (in the example SVAL1: 50 and SVAL2: 100). With these signal values, the foreground color of the progress bar changes. The appropriate signal colors are specified via parameters SC1 and SC2 (in the above example SC1: 9 (= yellow) and SC2: 7 (=red)).

The following applies for the specification of the limits for the progress bar:

MIN < SVAL1 < SVAL2 < MAX.

## Example 6b

Progress bar without color change:



Figure 5-2 Progress bar

| DEF PROGGY0 = (R/0,150///DT2,DO0//"$R[10]"//,,150/6,10) | | |
|---|---|---|
| Variable type: | | REAL |
| Limits / toggle field: | | |
| | MIN: | 0 |
| | MAX: | 150 |
| Attributes: | | |
| | Display type DT: | 2 (progress bar) |
| | Display option DO: | 0 (from left to right (default)) |
| System or user variable: | | $R[10] |
| Position of input/output field: | | |
| | Width: | 150 |
| Colors: | | |
| | Foreground color: | 6 (blue) |
| | Background color: | 10 (white) |

To use a progress bar without color change, the display mode DT (DisplayType) must be set to 2.

The orientation of the progress bar is determined via the attribute display option DO (DisplayOption) (see description for example 6a).

A MIN and a MAX value must be specified for the display of the progress bar (in the example MIN: 0, MAX: 150).

Depending on the current value of the variable PROGGY0, this setting displays a progress bar with the foreground color 6 (= blue) and the background color 10 (= white).

## 5.9 Example 7: Password input mode (asterisk)

### Example 7

To implement a field with hidden input, e.g. for the input of a password, the display mode DT must be set to 5. Asterisks are then displayed instead of the entered characters.



Figure 5-3 Password input mode (asterisk)

| DEF VAR_SET_PWD=(S//""//DT5) | | | |
|---|---|---|---|
| | Variable type: | | STRING |
| | Default setting: | | Empty string |
| | Attributes: | | |
| | | Display mode DT: | 5 (password input mode) |

## 5.10 Variable parameters

### Parameter overview

The following overview provides a brief explanation of the variable parameters. Subsequent chapters contain a more detailed description.

| Parameter | Description | |
|---|---|---|
| Variable type (Page 75) | The variable type must be specified. | |
| | R[x]: | REAL (+ digit for the decimal place) |
| | I: | INTEGER |
| | S[x]: | STRING (+ digit for string length) |
| | C: | CHARACTER (individual character) |
| | B: | BOOL |
| | V: | VARIANT |

| Parameter | Description | |
|---|---|---|
| Limits (Page 64) | Limit value MIN, limit value MAX<br>Default setting: Empty<br>The limit values are separated by a comma. Limits can be specified for types I, C and R in decimal formats or as characters in the form "A", "F". | |
| | Optionally, two signal colors SC1 and SC2 can be configured for the display of a progress bar with color change (variable attribute DT = 1) which are displayed as the respective foreground color of the bar when the signal value SVAL1 or SVAL2 is exceeded. The signal values are specified as integer values.<br>See application example for Progress bar (Page 66). | |
| Default setting (Page 79) | If no default setting has been configured and no system or user variable has been assigned to the variable, the first element of the toggle field is assigned. If no toggle field has been defined, there is no default setting, which means the status of the variable is "not calculated".<br>Default setting: No default | |
| Toggle field (Page 78) | List with predetermined entries in the IO field: The list is initiated by a *; the entries are separated by a comma. The entries can be assigned a value.<br>For the toggle field, the entry for the limit is interpreted as a list. If only one * is entered, a variable toggle field is created.<br>Default setting: None | |
| Texts (Page 63) | The sequence is specified. Instead of a short text, an image can also be displayed.<br>Default setting: Empty | |
| | Long text: | Text in the display line |
| | Short text: | Name of the dialog element |
| | Graphic text: | Text refers to the terms in the graphics |
| | Unit text: | Unit of the dialog element |
| | Tooltip | Serves as brief information in a screen form configuration for the display and toggle fields. The information is configured via plain text and language text ID. |

| Parameter | Description | |
|---|---|---|
| Attributes (Page 64) | The attributes influence the following properties:<br>• Display mode<br>• Display option<br>• Update rate<br>• Toggle symbol<br>• Tooltip<br>• Input mode<br>• Access level<br>• Alignment of short text<br>• Font size<br>• Limits<br>The attributes are separated by commas and appear in any order. A definition can be made for each component. | |
| | Display mode | dt0: Standard (input/output field or toggle field) (default)<br>dt1: Progress bar with color change<br>dt2: Progress bar without color change in the Operate style<br>dt4: List field<br>dt5: Password input mode (asterisk)<br>dt6: Input/output field with multi-line display |
| | Display option | Particularly, e.g. for the display modes dt1 and dt2 (progress bars), a display option may also have to be configured in combination.<br>do0: From left to right (default)<br>do1: From right to left<br>do2: From bottom to top<br>do3: From top to bottom |
| | Update rate | The attribute UR (update rate) controls the update of the display and therefore the processing of the associated, configured CHANGE block of variables or grid columns. Depending on the configuration, the CPU load can be drastically reduced and therefore shorter response times achieved on the user interface.<br>ur0: SlCap::standardUpdateRate() (currently = 200 ms, default value)<br>ur1: 50 ms<br>ur2: 100 ms<br>ur3: 200 ms<br>ur4: 500 ms<br>ur5: 1000 ms<br>ur6: 2000 ms<br>ur7: 5000 ms<br>ur8: 10000 ms |
| | Toggle symbol | tg0: Toggle symbol off (default)<br>tg1: Toggle symbol on<br>If the attribute TG is set to 1, the toggle symbol also appears in the tooltip of the input field.<br>Example: |

| Parameter | Description | |
|---|---|---|
| | | `DEF OFFS = (R//123.456/,,,,"My ToolTip"/TG1)` |
| | Tooltip | The tooltip text can also be changed during runtime via the variable property "TT".<br>Example:<br>`PRESS(VS1)`<br>`MyVar.TT = "My new ToolTip"`<br>`END_PRESS`<br>or<br>`DEF MyVar=(R///,,,,"My ToolTip-text")` |
| | Input mode | wr0: IO field invisible, short text visible<br>wr1: Read (no focus possible for input)<br>wr2: Read and write (line appears in white)<br>wr3: wr1 with focus<br>wr4: All variable elements invisible, no focus possible<br>wr5: The value entered is saved immediately on every keystroke (in contrast to wr2, where it is only saved when the field is exited or RETURN is pressed).<br>Default setting: wr2 |
| | Access level (Page 202) | Empty: Can always be written<br>ac0...ac7: Protection levels<br>If the access level is not adequate, then the first line is displayed in gray, default setting: ac7 |
| | Alignment of short text | al0: Left-justified<br>al1: Right-justified<br>al2: Centered<br>Default setting: al0 |
| | Font size | fs1: Default font size (8 pt.)<br>fs2: Double font size<br>Default setting: fs1<br>The clearances between the lines is defined. With the default font size, 16 lines will fit into the dialog. Graphics and unit text can only be configured in the default font size. |
| | Limits | Consequently, it is possible to check whether the values of the variable are within the MIN and MAX limits specified.<br>Default setting: Determined by specified limits<br>li0: No check<br>li1: Check with respect to min.<br>li2: Check with respect to max.<br>li3: Check with respect to min. and max. |
| | Behavior when opening | cb attributes specified for a variable in a variables definition take priority over the cb default setting in the dialog definition. Multiple attributes are separated by commas (see also Behavior when opening the dialog (attribute CB) (Page 204)). |
| | CB0:<br>CB1: | The CHANGE method is triggered when the screen is displayed if the variable has a valid value at this time (e.g. through default setting or NC/PLC variable). |

| Parameter | Description | |
|---|---|---|
| | | The CHANGE method is not explicitly triggered when the screen is displayed. If the variable has a configured NC/PLC variable, then the CHANGE method is of course still called. |
| Help display (Page 63) | Help display file: | Name of the png file<br>Default setting: Empty |
| | The name of the Help display file appears in double quotation marks. The display appears automatically (instead of the previous graphic) if the cursor is positioned on this variable. | |
| System or user variable (Page 65) | System or user data from the NC/PLC can be assigned to the variable. The system or user variable appears in double quotation marks.<br>Reference: List Manual System Variables, /PGAsl/ | |
| Position of short text (Page 80) | Position of short text (distance from left, distance from top, width)<br>The positions are entered in pixels and relate to the upper left-hand corner of the main body of the dialog. The entries are separated by commas. | |
| Position of input/output field (Page 80) | Position of input/output field (distance from left, distance from top, width, height)<br>The positions are entered in pixels and relate to the upper left-hand corner of the main body of the dialog. The entries are separated by commas. If this position changes, the positions of the short text, graphic text and unit text also change. | |
| Colors (Page 63) | Foreground and background colors for input/output fields, short texts, graphic texts, unit texts and signal colors for progress bars:<br>The colors are separated by a comma.<br>For specification of the color, see Chapter List of colors (Page 202).<br>Default setting for input/output field: Foreground color: Black, background color: White<br>The default colors of the input/output field depend on the "wr" write mode:<br>Default setting for short text, graphic text, unit text: Foreground color: Black, background color: Transparent. | |
| | Optionally, two signal colors SC1 and SC2 can be configured for the display of a progress bar with color change (variable attribute DT = 1) which are displayed as the respective foreground color of the bar when the signal value SVAL1 or SVAL2 is exceeded.<br>See application example for Progress bar (Page 66). | |
| | The colors are expected in the following order in the variables definition:<br>1. Foreground color of the input/output field: FC<br>2. Background color of the input/output field: BC<br>3. Foreground color of the short text: FC_ST<br>4. Background color of the short text: BC_ST<br>5. Foreground color of the graphic text: FC_GT<br>6. Background color of the graphic text: BC_GT<br>7. Foreground color of the unit text: FC_UT<br>8. Background color of the unit text: BC_UT<br>9. Signal color 1<br>10. Signal color 2 | |

| Parameter | Description |
|---|---|
| Online help file (Page 58) | The name of the online help file appears in double quotation marks. |
| I/O field with integrated unit selection | You can use the input/output fields with integrated unit selection to toggle between different units. If the cursor is moved to the input field, the integrated unit selection is highlighted (focus does not have to be explicitly placed on it). A tooltip with a toggle symbol is also displayed with the appropriate reference to this functionality.<br>Examples:<br>`DEF VarEdit=(R/////////200,,100///"VarTgl")`<br>`VarTgl=(S/*0="mm",1="inch"/0//WR2////302,,40)`<br>or<br>`DEF VarEdit_2={TYP="R", VAL=1.234, X=200, W=100, LINK_TGL="vartgl_2"},`<br>`VARTgl_2={TYP="S", TGL="* 0=""mm"", 1=""inch""",WR=2, X=302, W=40}` |

## Variable: Changing properties

The variables are assigned a new value in the notation *Identifier.Property = Value* when a change is made. The expression to the right of the equality sign is evaluated and assigned to the variable or variable property.

| | |
|---|---|
| Identifier.**ac** = Access level | (ac: access level) |
| Identifier.**al** = Text alignment | (al: alignment) |
| Identifier.**bc** = Background color of the input/output field | (bc: back color) |
| Identifier.**bc_gt** = Background color of the graphic text | (bc: back color)<br>(gt: graphic text) |
| Identifier.**bc_st** = Background color of the short text | (bc: back color)<br>(st: short text) |
| Identifier.**bc_ut** = Background color of the unit text | (bc: back color)<br>(ut: unit text) |
| Identifier.**do** = Display option | (do: display option) |
| Identifier.**dt** = Display mode | (dt: display type) |
| Identifier.**fc** = Foreground color of the input/output field | (fc: front color) |
| Identifier.**fc_gt** = Foreground color of the graphic text | (fc: front color)<br>(gt: graphic text) |
| Identifier.**fc_st** = Foreground color of the short text | (fc: front color)<br>(st: short text) |
| Identifier.**fc_ut** = Foreground color of the unit text | (fc: front color)<br>(ut: unit text) |
| Identifier.**fs** = Font size | (fs: font size) |
| Identifier.**gt** = Graphic text | (gt: graphic text) |
| Identifier.**hlp** = Help display | (hlp: help) |
| Identifier.**li** = Limit | (li: limit) |
| Identifier.**lt** = Long text | (lt: long text) |
| Identifier.**max** = MAX limits | (max: maximum) |
| Identifier.**min** = MIN limits | (min: minimum) |

| | |
|---|---|
| Identifier.**sc** = Signal color | (sc: signal color) |
| Identifier.**st** = Short text | (st: short text) |
| Identifier.**tg** = Toggle symbol | (tg: toggle) |
| Identifier.**tt** = Tooltip | (tt: tooltip |
| Identifier .**typ** = Variable type | (typ: type) |
| Identifier.**ur** = Update rate | (ur: update rate) |
| Identifier.**ut** = Unit text | (ut: unit text) |
| Identifier.**val** = Variable value | (val: value) |
| Identifier.**var** = System or user variable | (var: variable) |
| Identifier.**vld** = Variable status | (vld: validation) |
| Identifier.**wr** = Input mode | (wr: write) |

### See also

Extended configuration syntax (Page 36)

## 5.11 Details on the variable type

### Variable type INTEGER

The following extensions for determining the display in the input/output field and the memory utilization are possible for the "INTEGER" type:

**2nd character** in the extension data type

| **Display format** | |
|---|---|
| B | Binary |
| D | Decimal signed |
| H | Hexadecimal |
| No data | Decimal signed |

**3rd and/or 4th character** in the extension data type

| **Memory utilization** | |
|---|---|
| B | Byte |
| W | Word |
| D | Double word |
| BU | Byte, Unsigned |
| WU | Word, Unsigned |
| DU | Double word, Unsigned |

## Sequence of characters in the INTEGER data type

1. "I" Basic INTEGER designation
2. Display format
3. Memory utilization
4. "U" Unsigned

| Valid INTEGER type specifications: | |
|---|---|
| IB | Integer variable 32 bits in binary notation |
| IBD | Integer variable 32 bits in binary notation |
| IBW | Integer variable 16 bits in binary notation |
| IBB | Integer variable 8 bits in binary notation |
| I | Integer variable 32 bits in decimal notation signed |
| IDD | Integer variable 32 bits in decimal notation signed |
| IDW | Integer variable 16 bits in decimal notation signed |
| IDB | Integer variable 8 bits in decimal notation signed |
| IDDU | Integer variable 32 bits in decimal notation unsigned |
| IDWU | Integer variable 16 bits in decimal notation unsigned |
| IDBU | Integer variable 8 bits in decimal notation unsigned |
| IH | Integer variable 32 bits in hexadecimal notation |
| IHDU | Integer variable 32 bits in hexadecimal notation |
| IHWU | Integer variable 16 bits in hexadecimal notation |
| IHBU | Integer variable 8 bits in hexadecimal notation |

## VARIANT variable type

The VARIANT variable type is determined by the data type of the last value assignment. If the assigned or entered value starts with '-', '+', '.' or a number ('0'-'9'), then the value is interpreted as numeric. In all other cases as a string.

It can be scanned using the ISNUM or ISSTR functions. The VARIANT type is mainly suited to the purpose of writing either variable names or numerical values to the NC code.

## Programming

The data type of variables can be checked:

| Syntax: | **ISNUM** *(VAR)* | |
|---|---|---|
| Parameter: | VAR | Name of the variable whose data type is to be checked. |
| | | The result of the scan can be: |
| | FALSE = | not a numerical variable (data type = STRING) |
| | TRUE = | numerical variable (data type = REAL) |

| Syntax: | **ISSTR** *(VAR)* | |
|---|---|---|
| Parameter: | VAR | Name of the variable whose data type is to be checked. |
| | | The result of the scan can be: |
| | FALSE = | numerical variable (data type = REAL) |
| | TRUE = | not a numerical variable (data type = STRING) |
| Example: | `IF ISNUM(VAR1) == TRUE`<br>`IF ISSTR(REG[4]+2) == TRUE` | |

The display mode of variables can be changed:

- For INTEGER, the display type can be changed.

| | |
|---|---|
| B | Binary |
| D | Decimal signed |
| H | Hexadecimal |
| unsigned | |
| With the addition of U for Unsigned | |

- For REAL data types, only the number of places after the decimal point can be changed. Changing the type is illegal and generates an error message in the easyscreen_log.txt. file. Example:
  ```
  Var1.typ = "IBW"
  Var2.typ = "R3"
  ```

## Numerical formats

Numbers can be represented in either binary, decimal, hexadecimal or exponential notation:

| | |
|---|---|
| Binary | B01110110 |
| Decimal | 123.45 |
| Hexadecimal | HF1A9 |
| exponential | -1.23EX-3 |
| Examples: | |
| | `VAR1 = HF1A9` |
| | `REG[0]= B01110110` |
| | `DEF VAR7 = (R//-1.23EX-3)` |

**Note**

When codes are generated with the "GC" function, only numerical values in decimal or exponential notation are evaluated, but **not** those in binary or hexadecimal notation.

## See also

Variable parameters (Page 69)

# 5.12 Details on the toggle field

## Description

The toggle field extension function displays texts (entries in toggle field) as a function of NC/PLC variables. A variable, which makes use of a toggle field extension, is read-only. The list of the toggle field is opened by pressing the ENTER key.

## Programming

| | | |
|---|---|---|
| Syntax: | **DEF VAR1=(IB/+ $85000/15////"DB90.DBB5")** or<br>**DEF VAR_TGL = (S/* "Hello", "Run", "MyScreens"/"Run")** | |
| Description: | When the dialog is opened, the content of text number $85015 is displayed in the IO field. Default value 15 is entered in system variable DB90.DBB5. If the value saved in system variable DB90.DBB5 changes, the displayed text number $(85000 + <DB90.DBB5>) is recalculated in response to every change. | |
| Parameter: | Variable type | Type of variables specified in the system or user variable. |
| | Text number | Number (basis) of the language-specific text valid as the basis number. |
| | System or user variable | System or user variable (offset) via which the final text number (basis + offset) is displayed. |

## Toggle-field-dependent displays

The toggle field is overlaid with graphics, which change depending on the value of the memory byte. If the value of the memory byte is 1, "image1.png" will appear. If it is 2, "image2.png" will appear.

```
DEF VAR1=(IDB/*1="\\image1.png", 2="\\image2.png"//,$85000/wr1//"MB[130]"//160,40,50,50)
```

The size and position of the image is defined under "Position of IO field (left, top, width, height)".

## Virtual toggle key

There is no list for toggle fields without configured list, e.g. `DEF NoTglList=(R/*)`. The next element is not generated until the toggle key is actuated or the associated CHANGE() method of the corresponding variable run through.

In this case, a small virtual keyboard (comprising only a toggle key) is displayed on the right next to the variable toggle field for operation with a touch panel.

The display of the virtual toggle key can also be forced via the following entry in the configuration file slguiconfig.ini irrespective of a touch panel.

```
[VirtualKeyboard]
```

```
; Forces easyscreen virtual toggle keyboard function
ForceEasyscreenVirtualToggleKey = true
```

### See also

Variable parameters (Page 69)

## 5.13 Details on the default setting

### Overview

A variable can assume various states depending on whether a default value, or a system or user variable, or both, is assigned to the variable field (input/output field or toggle field), (not calculated: toggling is only possible if a valid value has been assigned to the variable).

### Scope of the default settings

| Condition | | | Reaction of field type |
|---|---|---|---|
| Field type | Default setting | System or user variable | |
| I/O field | Yes | Yes | Write default value to system or user variable |
| | No | Yes | Use system or user variable as default value |
| | Fault | Yes | Not calculated, system or user variable is not written/ used. |
| | Yes | No | Default setting |
| | No | No | Not calculated |
| | Fault | No | Not calculated |
| | Yes | Fault | Not calculated |
| | No | Fault | Not calculated |
| | Fault | Fault | Not calculated |
| Toggle | Yes | Yes | Write default value to system or user variable |
| | No | Yes | Use system or user variable as default value |
| | Fault | Yes | Not calculated,<br>system or user variable not written/used |
| | Yes | No | Default setting |
| | No | No | Default = first toggle field element |
| | Fault | No | Not calculated |
| | Yes | Fault | Not calculated |
| | No | Fault | Not calculated |
| | Fault | Fault | Not calculated |

### See also

Variable parameters (Page 69)

## 5.14 Details on the position of the short text, position of the input/output field

### Overview

The short text and graphic text, as well as the input/output field and unit text, are each treated like a unit, i.e., position settings for short text apply to the graphic text and settings for the input/output field and to unit text.

### Programming

The configured position entry overwrites the default value, i.e., only one value can be changed. If no position settings have been configured for subsequent screen form elements, then the position settings for the preceding screen form element are applied.

If no positions have been specified for any dialog elements, the default setting is applied. By default, the column width for the short text and input/output field is calculated for each line based on the number of columns and maximum line width, i.e.,
column width = maximum line width/number of columns.

The width of the graphics and unit text is predefined and optimized to suit the requirements of programming support. If graphics or unit text has been configured, the width of the short text or I/O field is reduced accordingly.

The order of short text and I/O field can be reversed by position settings.

### See also

Variable parameters (Page 69)

## 5.15 Use of strings

### Strings

Strings can be used as part of the configuration. These allow text to be displayed dynamically or different texts to be chained for the purpose of code generation.

### Rules

The following rules must be observed with regard to string variables:

- Logic operations are processed from left to right.
- Nested expressions are solved from the inside outwards.

- No distinction is made between uppercase and lowercase type.
- String variables are generally displayed left justified.

Strings can be deleted simply by assigning a blank string.

Strings can be appended after the equality sign using the operator "<<". Quotation marks (") in the string are represented by two successive quotation mark symbols. Strings can be checked for equality in IF instructions.

## Example

Default settings for the following examples:

```
VAR1.VAL = "This is an"

VAR8.VAL = 4

VAR14.VAL = 15

VAR2.VAL = "Error"

$85001 = "This is an"

$85002 = "Alarm text"
```

Editing strings:

- Chaining of strings:
  ```
  VAR12.VAL = VAR1 << " Error."   ;Result: "This is an error"
  ```
- Deleting a variable:
  ```
  VAR10.VAL = ""   ;Result: Blank string
  ```
- Setting a variable with a text variable:
  ```
  VAR11.VAL = VAR1.VAL   ;Result: "This is an"
  ```
- Data type matching:
  ```
  VAR13.VAL ="This is the " << (VAR14 - VAR8) << ". error"
              ;Result: "This is the 11th error"
  ```
- Treatment of numerical values:
  ```
  VAR13.VAL = "Error" << VAR14.VAL << ": " << $85001 << $85002
           ;Result: "Error 15: "This is an alarm text"
  IF VAR15 == "Error"   ;Strings in IF statement
    VAR16 = 18.1234
           ;Result: VAR16 equals 18.1234,
           ;if VAR15 equals "Error".
  ENDIF
  ```
- Quotation marks within a string:
  ```
  VAR2="Hello, this is a " Test""
     ;Result: Hello, this is a " Test"
  ```
- System or user-variable strings dependent on variable content:
  ```
  VAR2.Var = "$R[" << VAR8 << "]"   ;Result: $R[4]
  ```

## See also

STRING functions (Page 151)

## 5.16 CURPOS variable

### Description

The CURPOS variable calls or manipulates the position of the cursor in the active input field of the current dialog. The variable shows how many characters are in front of the cursor. If the cursor is located at the start of the input field, then CURPOS assumes the value of 0. If the value of CURPOS is changed, then the cursor is positioned at the appropriate location in the input field.

In order to be able to respond to changes in the variable value, it is possible to monitor for changes using a CHANGE method. If the value of CURPOS changes, then a jump is made to the CHANGE method and the instructions contained there are executed.

## 5.17 CURVER variable

### Description

The CURVER (CURrent VERsion) property allows the programming to be adapted in order to handle different versions. The CURVER variable is read-only.

**Note**

Even if previously recompiled with an older version, the code is automatically generated with the most recent version. The "GC" command always generates the most recent version. An additional identifier indicating the generated version is inserted in the user comment of the generated code in versions > 0.

### Rules

The most recent dialog with all its variables is always displayed.

- Variables used previously may not be changed.
- New variables are inserted in the existing (cycle) programming in arbitrary order.
- It is not permissible to delete variables from a dialog from one version to the next.
- The dialog must contain all variables of all versions.

### Example

```
(IF CURVER==1 ...)                ; When the code is recompiled, CURVER is automat-
                                    ically assigned the version of the recompiled
                                    code.
```

## 5.18 ENTRY variable

### Description

The ENTRY variable can be used to check by what method a dialog has been called.

### Programming

| Syntax: | **ENTRY** | |
|---|---|---|
| Description: | The ENTRY variable is a read only variable. | |
| Return Value: | | The result of the scan can be: |
| | 0 = | No programming support |
| | 1 = | Start of a screen via softkey; no code has been generated yet (default as configured) |
| | 2 = | Start of a screen via softkey; code has been generated<br>(default from the code generated last by this screen) |
| | 3 = | Recompilation with user comment (# lines) |
| | 4 = | Code_type = 0: NC code with user comment (# lines) |
| | 5 = | Code_type = 1: NC code without user comment (# lines) |

### Example

```
IF ENTRY == 0
  DLGL("The dialog was not called during programming")
ELSE
  DLGL("The dialog was called during programming")
ENDIF
```

## 5.19 ERR variable

### Description

Variable ERR checks whether the preceding line has been executed correctly.

### Programming

| Syntax: | **ERR** | |
|---|---|---|
| Description: | The ERR variable is read-only. | |
| Return value: | | The result of the scan can be: |
| | FALSE = | previous line was executed error-free |
| | TRUE = | previous line was not executed error-free |

### Example

```
VAR4 = Thread[VAR1,"CDM",3]            ; Output value from array
IF ERR == TRUE                         ; Query whether the value has been found in
                                       the array
  VAR5 = "Error accessing array"
                                       ; If the value has not been found in the ar-
                                       ray, the value "Error accessing array" is as-
                                       signed to the variables.
ELSE
  VAR5 = "All OK"                      ; If the value has been found in the array,
                                       the value "All OK" is assigned to the varia-
                                       bles.
ENDIF
```

## 5.20 FILE_ERR variable

### Description

Variable FILE_ERR can be used to check whether the preceding GC or CP command has been executed correctly.

### Programming

| Syntax: | **FILE_ERR** | |
|---|---|---|
| Description: | The FILE_ERR variable is read-only. | |
| Return value: | | Possible results are: |
| | 0 = | Operation okay |
| | 1 = | Drive/path not available |
| | 2 = | Path/file access error |
| | 3 = | Drive not ready |
| | 4 = | Incorrect file name |
| | 5 = | File is already open |
| | 6 = | Access denied |
| | 7 = | Target path not available or not permitted |
| | 8 = | Copy source same as target |
| | 10 = | Internal error: FILE_ERR = 10 means that the error cannot be classified in the other categories. |

### Example

```
CP("D:\source.mpf","E:\target.mpf")
                                                ; Copy from source.mpf to E:\tar-
                                                get.mpf
IF FILE_ERR > 0                                 ; Query whether error has occurred
   IF FILE_ERR == 1                             ; Query specific error numbers and
                                                output associated error text
      VAR5 = "Drive/path not available"
   ELSE
      IF FILE_ERR == 2
         VAR5 = "Path/file access error"
      ELSE
         IF FILE_ERR == 3
            VAR5 = "incorrect file name"
         ENDIF
      ENDIF
   ENDIF
ELSE
   VAR5 = "All OK"                              ; If no errors have occurred in CP
                                                (or GC), "All OK" is output
ENDIF
```

## 5.21 FOC variable

### Description

With the variable FOC, the input focus (current active input/output field) is controlled in a dialog. The reaction of the cursor left, right, up, down as well as PGUP, PGDN are predefined.

**Note**

The FOC function must not be initiated as a result of a navigation event. The cursor position may only be changed in softkey PRESS methods, CHANGE methods, etc.

The FOC function cannot be applied to variables with input mode wr = 0 and wr = 4 or to Help variables.

### Programming

| Syntax: | **FOC** |
|---|---|
| Description: | The variable can be read and written. |

| | | |
|---|---|---|
| Return value: | Read | The result is the name of the variable to which the FOC function has been applied. |
| | Write | It is possible to assign either a string or a numerical value. A string is interpreted as a variable name and a numerical value as a variable index. |

### Example

```
IF FOC == "Var1"                ; Read focus
  REG[1] = Var1
ELSE
  REG[1] = Var2
ENDIF

FOC = "Var1"                    ; The input focus is assigned to variable 1.
FOC = 3                         ; The input focus is assigned to the 3rd dialog element with WR ≥ 2.
```

### See also

FOCUS (Page 99)

## 5.22 Variable S_ALEVEL

### Description

The current access level can be queried in the configuration with the screen property S_ALEVEL.

### Programming

| | |
|---|---|
| Syntax: | **S_ALEVEL** |
| Description: | Query of the current access level |
| Return value: | 0: System |
| | 1: Manufacturer |
| | 2: Service |
| | 3: User |
| | 4: Key switch 3 |
| | 5: Key switch 2 |
| | 6: Key switch 1 |
| | 7: Key switch 0 |

### Example

```
REG[0] = S_ALEVEL
```

### See also

ACCESSLEVEL (Page 96)

## 5.23 S_CHAN variable

### Description

The S_CHAN variable determines the number of the current channel for display or evaluation purposes.

### Programming

| | |
|---|---|
| Syntax: | **S_CHAN** |
| Description: | Query of the current channel number |
| Return value: | Channel number |

### Example

```
REG[0] = S_CHAN
```

### See also

CHANNEL (Page 98)

## 5.24 Variable S_CONTROL

### Description

The current control name can be queried in the configuration with the screen property S_CONTROL.

### Programming

| | |
|---|---|
| Syntax: | **S_CONTROL** |
| Description: | Query of the current control name |
| Return value: | The control name is the section name in the mmc.ini |

### Example

```
REG[0] = S_CONTROL
```

### See also

CONTROL (Page 98)

## 5.25 Variable S_LANG

### Description

The current language can be queried in the configuration with the screen property S_LANG.

### Programming

| | |
|---|---|
| Syntax: | **S_LANG** |
| Description: | Query of the current language |
| Return value: | Language code from resources.xml, e.g. "deu", "eng", etc. |

### Example

```
REG[0] = S_LANG
```

### See also

LANGUAGE (Page 100)

## 5.26 Variable S_NCCODEREADONLY

### Description

The screen property S_NCCODEREADONLY is only relevant when a cycle is recompiled in the editor with a "Run MyScreens" dialog. S_NCCODEREADONLY determines whether a recompiled NC code from the editor can be changed or not.

The following applies for the return value:

- TRUE: The recompiled NC code from the editor can be changed
- FALSE: The recompiled NC code from the editor cannot be changed (read-only), because, for example, it is already in the preprocessing (= TRUE).

## 5.27 Variables S_RESX and S_RESY

### Description

The current resolution or its X and Y component can be queried in the configuration with the screen properties S_RESX and S_RESY.

### Example

```
REG[0] = S_RESY
```

### See also

RESOLUTION (Page 105)

# Programming commands

# 6

## 6.1 Operators

### Overview

The following operators can be used when programming:

- Mathematical operators
- Relational operators
- Logic (Boolean) operators
- Bit operators
- Trigonometric functions

## 6.1.1 Mathematical operators

### Overview

| Mathematical operators | Identifier |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| MOD | Modulo operation |
| ( ) | Parentheses |
| AND | AND operator |
| OR | OR operator |
| NOT | NOT operator |
| ROUND | Round off numbers with decimal places |

**Example:** `VAR1.VAL = 45 * (4 + 3)`

### ROUND

The ROUND operator is used to round off numbers with up to 12 decimal places during execution of a dialog configuration. The variable fields cannot accept the decimal places in the display.

## Use

ROUND is controlled by the user with two parameters:

```
VAR1 = 5.2328543
VAR2 = ROUND( VAR1, 4 )
```

**Result:** `VAR2 = 5.2339`

VAR1 contains the number to be rounded. The parameter “4” indicates the number of decimal places in the result, which is placed in VAR2.

## Trigonometric functions

| Trigonometric functions | Identifier |
|---|---|
| SIN(x) | Sine of x |
| COS(x) | Cosine of x |
| TAN(x) | Tangent of x |
| ATAN(x, y) | Arc tangent of x/y |
| SQRT(x) | Square root of x |
| ABS(x) | Absolute value of x |
| SDEG(x) | Conversion to degrees |
| SRAD(x) | Conversion to radian |
| CALC_ASIN(x) | Arc sine of x |
| CALC_ACOS(x) | Arc cosine of x |

**Note**

The functions operate with radian measure. The functions SDEG() and SRAD() can be used for conversion.

**Example:** `VAR1.VAL = SQRT(2)`

## Mathematical functions

| Mathematical functions | Identifier |
|---|---|
| CALC_CEIL(x) | Determines the next highest integer of x (round up) |
| CALC_FLOOR(x) | Determines the next lowest integer of x (round down) |
| CALC_LOG(x) | Determines the (natural) logarithm for the base e of x |
| CALC_LOG10(x) | Determines the logarithm for the base 10 of x |
| CALC_POW(x, y) | Determines x to the power of y |
| CALC_MIN(x, y) | Determines the lower number of x and y |
| CALC_MAX(x, y) | Determines the higher number of x and y |

## Random function: Random number

| Syntax: | **RANDOM** *(lower limit value, upper limit value)* | |
|---|---|---|
| Description: | The **RANDOM** function returns a pseudo random number in a specified range. | |
| Parameters: | Lower limit value | Lower limit value >= -32767,<br>Lower limit value < upper limit value |
| | Upper limit value | Upper limit value <= 32767 |

## Example

```
REG[0] = RANDOM(-10,10)         ; Possible result = -3
```

## *Constants*

| *Constants* | |
|---|---|
| *PI* | *3.14159265358979323846* |
| *FALSE* | *0* |
| *TRUE* | *1* |

*Example:* `VAR1.VAL = PI`

## *Comparison operators*

| *Comparison operators* | |
|---|---|
| *==* | *Equal to* |
| *<>* | *Not equal to* |
| *>* | *Greater than* |
| *<* | *Less than* |
| *>=* | *Greater than or equal to* |
| *<=* | *Less than or equal to* |

*Example:*

```
IF VAR1.VAL == 1
  VAR2.VAL = TRUE
ENDIF
```

### *Conditions*

*The nesting depth is unlimited.*

*Condition with one command:*

```
IF
...
ENDIF
```

*Condition with two commands:*

```
IF
...
ELSE
...
ENDIF
```

## 6.1.2 Bit operators

### Overview

| Bit operators | Identifier |
|---|---|
| BOR | Bit-serial OR |
| BXOR | Bit-serial XOR |
| BAND | Bit-serial AND |
| BNOT | Bit-serial NOT |
| SHL | Shift bits to left |
| SHR | Shift bits to right |

### SHL operator

Bits are shifted to the left using the SHL (SHIFT LEFT) operator. You can specify both the value to be shifted and the number of shift increments directly or via a variable. If the limit of the data format is reached, the bits are shifted beyond the limit without displaying an error message.

### Use

| | | |
|---|---|---|
| Syntax: | variable = *value* **SHL***increment* | |
| Description: | Shift Left | |
| Parameters: | value | value to be shifted |
| | increment | number of shift increments |

### Example

```
PRESS(VS1)
```

```
  VAR01 = 16 SHL 2           ; Result = 64
  VAR02 = VAR02 SHL VAR04    ; Convert content of VAR02 to 32-bit unsigned
                             , and shift content to left by number of bits specified
                             in VAR04. Then convert 32-bit value back to format of
                             variable VAR02.
END_PRESS
```

### SHR operator

Bits are shifted to the RIGHT using the SHR (SHIFT RIGHT) function. You can specify both the value to be shifted and the number of shift increments directly or via a variable. If the limit of the data format is reached, the bits are shifted beyond the limit without displaying an error message.

### Use

| | | | |
|---|---|---|---|
| Syntax: | variable = *value* **SHR***increment* | | |
| Description: | Shift Right | | |
| Parameters: | value | value to be shifted | |
| | increment | number of shift increments | |

### Example

```
PRESS(VS1)
  VAR01 = 16 SHR 2               ; Result = 4
  VAR02 = VAR02 SHR VAR04        ; Convert content of VAR02 to 32-bit unsigned
                                 , and shift content to right by number of bits
                                 specified in VAR04. Then convert 32-bit value back
                                 to format of variable VAR02.
END_PRESS
```

## 6.2 Methods

### Overview

Various types of event (exit input field, actuate softkey) can initiate specific actions in dialogs and dialog-dependent softkey menus (softkey menus that are called from a newly configured dialog). These actions are configured in methods.

The following table shows the basic principle used to program a method:

| Definition block | Comment | Chapter reference |
|---|---|---|
| `PRESS(HS1)` | ;Method start identifier | |
| `LM...`<br>`  LS...` | ;Functions | See chapter "Functions" |
| `Var1.st = ...` | ;Changing properties | see chapter "Softkey menu" and chapter "Dialog elements" |
| `Var2 = Var3 + Var4`<br>`...`<br>`EXIT` | ;Calculation with variables | See chapter "Defining variables" |
| `END_PRESS` | ;Method end identifier | |

## 6.2.1 ACCESSLEVEL

### Description

The ACCESSLEVEL method is run through when the current access level has changed for an opened screen.

### Programming

| | |
|---|---|
| Syntax: | ACCESSLEVEL<br><instructions><br>END_ACCESSLEVEL |
| Description: | Access level |
| Parameters: | - None - |

### See also

Variable S_ALEVEL (Page 86)

## 6.2.2 CHANGE

### Description

CHANGE methods are executed if a variable value changes, i.e., variable calculations that are performed as soon as a variable value changes are configured within a CHANGE method.

There are two types of CHANGE method, i.e., element-specific and global:

- The **element-specific CHANGE method** is executed if the value of a specified variable changes. If a system or user variable is assigned to a variable, cyclic updating of the variable value can be configured in a CHANGE method.
- The **global CHANGE method** is executed if the value of any variable changes and no element-specific CHANGE method has been configured.

## "Element-specific" programming

| | | |
|---|---|---|
| Syntax: | CHANGE*(Identifier)*<br>...<br>END_CHANGE | |
| Description: | Changes the value of a specific variable | |
| Parameters: | Identifier | Name of the variable |

## Example

```
DEF VAR1=(I//////"DB20.DBB1")        ; A system variable is assigned to Var1
CHANGE(VAR1)
  IF VAR1.Val <> 1
  VAR1.st="Tool OK!"                 ; If the value of the system variable ≠ 1, the
                                     short text of the variable states: Tool OK!
    otto=1
  ELSE
    VAR1.st="Attention: Error!"      ; If the value of the system variable = 1, the
                                     short text of the variable states: Attention:
                                     Error!
    otto=2
  ENDIF
  VAR2.Var=2
END_CHANGE
```

## "Global" programming

| | |
|---|---|
| Syntax: | CHANGE()<br>...<br>END_CHANGE |
| Description: | Changes any variable value |
| Parameters: | - None - |

### Example

```
CHANGE()
  EXIT                                ; If any of the variable values change, the di-
                                      alog will be terminated.
END_CHANGE
```

### See also

Behavior when opening the dialog (attribute CB) (Page 204)

## 6.2.3 CHANNEL

### Description

The CHANNEL method is run through when the current channel has changed for an opened screen, i.e. a channel switchover has been performed.

### Programming

| | |
|---|---|
| Syntax: | CHANNEL<br><instructions><br>END_CHANNEL |
| Description: | Channel switchover |
| Parameters: | - None - |

### See also

S_CHAN variable (Page 87)

## 6.2.4 CONTROL

### Description

The CONTROL method is run through when the current control has changed for an opened screen, i.e. typically at a 1:n switchover.

### Programming

| | |
|---|---|
| Syntax: | CONTROL<br><instructions><br>END_CONTROL |
| Description: | Control switchover |
| Parameters: | - None - |

### See also

Variable S_CONTROL (Page 87)

## 6.2.5 FOCUS

### Description

The FOCUS method is executed if the focus (cursor) is positioned on another field in the dialog.

The FOCUS method must not be initiated as a result of a navigation event. The cursor position may only be changed in softkey PRESS methods, CHANGE methods, etc. The response of cursor movements is predefined.

**Note**

Within the FOCUS method, it is not possible to select a different variable, nor can a new dialog be loaded.

### Programming

| | |
|---|---|
| Syntax: | FOCUS<br>...<br>END_FOCUS |
| Description: | Positions the cursor |
| Parameters: | - None - |

### Example

```
FOCUS
  DLGL("The focus has been placed on variable << FOC << ".)
END_FOCUS
```

### See also

FOC variable (Page 85)

## 6.2.6 LANGUAGE

### Description

The LANGUAGE method is run through when the current language has changed for an opened screen.

### Programming

| | |
|---|---|
| Syntax: | LANGUAGE<br><instructions><br>END_LANGUAGE |
| Description: | Language |
| Parameters: | - None - |

### See also

Variable S_LANG (Page 88)

## 6.2.7 LOAD

### Description

The LOAD method is executed after the variable and softkey definitions (DEF Var1= ..., HS1= ...) have been interpreted. At this time, the dialog is not yet displayed.

### Programming

| | |
|---|---|
| Syntax: | LOAD<br>...<br>END_LOAD |
| Description: | Download |
| Parameters: | - None - |

### Example

```
LOAD                          ; Start identifier
```

```
  Screen form1.Hd =  $85111  ; Assign text for dialog header from language file
  VAR1.Min = 0               ; Assign MIN variable limit
  VAR1.Max = 1000            ; Assign MAX variable limit
END_LOAD                     ; End code
```

### See also

Line, dividing line, rectangle, circle and ellipse (Page 163)

## 6.2.8 UNLOAD

### Description

The UNLOAD method is executed before a dialog is unloaded.

### Programming

| | |
|---|---|
| Syntax: | UNLOAD<br>...<br>END_UNLOAD |
| Description: | Unload |
| Parameters: | - None - |

### Example

```
UNLOAD
  REG[1] = VAR1            ; Save variable in register
END_UNLOAD
```

## 6.2.9 OUTPUT

### Description

The OUTPUT method is executed if the "GC" function is called. Variables and Help variables are configured as an NC code in an OUTPUT method. The individual elements in a code line are linked by means of blanks.

**Note**

The NC code can be generated in an extra file by means of file functions and transferred to the NC.

### Programming

| | | |
|---|---|---|
| Syntax: | OUTPUT *(Identifier)*<br>...<br>END_OUTPUT | |
| Description: | Outputs variables in the NC program. | |
| Parameters: | Identifier | Name of OUTPUT method |

### Block numbers and skip identifiers

The OUTPUT method must not contain line numbers or skip identifiers if you wish to keep the line numbers and skip identifiers directly set with active program support in the parts program in case of recompilations.

Editor changes in the parts program produce the following response:

| Condition | Response |
|---|---|
| Number of blocks remains unchanged. | Block numbers are retained. |
| Number of blocks is reduced. | The highest block numbers are canceled. |
| Number of blocks is increased. | New blocks are not numbered. |

### Example

```
OUTPUT(CODE1)
  "CYCLE82(" Var1.val "," Var2.val "," Var3.val ","Var4.val "," Var5.val "," Var6.val ")"
END_OUTPUT
```

## 6.2.10 PRESS

### Description

The PRESS method is executed when the corresponding softkey is pressed.

### Programming

| | | | |
|---|---|---|---|
| Syntax: | PRESS*(softkey)*<br>...<br>END_PRESS | | |
| Designation: | Pressing a softkey | | |
| Parameter: | Softkey | Name of softkey: HS1 - HS8 and VS1 - VS8 | |
| | RECALL | <RECALL> key | |
| | ENTER | For the <ENTER> key, see PRESS(ENTER) (Page 104) | |
| | TOGGLE | For the <TOGGLE> key, see PRESS(TOGGLE) (Page 104) | |
| | PU | Page Up | Screen up |
| | PD | Page Down | Screen down |
| | SL | Scroll left | Cursor left |
| | SR | Scroll right | Cursor right |
| | SU | Scroll up | Cursor up |
| | SD | Scroll down | Cursor down |

### Example

```
HS1 = ("another softkey menu")
HS2=("no function")
PRESS(HS1)
  LS("Menu1")                    ; Load another softkey menu
  Var2 = Var3 + Var1
END_PRESS
PRESS (HS2)
END_PRESS
PRESS(PU)
  INDEX = INDEX -7
  CALL("UP1")
END_PRESS
```

## 6.2.11 PRESS(ENTER)

### Description

The PRESS(ENTER) method is always called when the Enter key is pressed for a variable with input/output field with input mode WR3 or WR5:

- WR3: Navigation to field and pressing of the Enter key
- WR5: In the input mode, transfer of the value with the Enter key

### Programming

| | |
|---|---|
| Syntax: | PRESS(ENTER)<br><instructions><br>END_PRESS |
| Description: | Enter key pressed |
| Parameters: | - None - |

## 6.2.12 PRESS(TOGGLE)

### Description

The PRESS(TOGGLE) method is always called when the toggle key is pressed irrespective of the currently focused variable.

When required, the FOC screen property can be used to determine which variable is currently in focus.

### Programming

| | |
|---|---|
| Syntax: | PRESS(TOGGLE)<br><instructions><br>END_PRESS |
| Description: | Toggle key pressed |
| Parameter: | - None - |

### Example

```
PRESS(TOGGLE)
   DLGL("Toggle key pressed at variable " << FOC)        ; The FOC screen property determines which
                                                         variable is currently in focus
END_PRESS
```

## 6.2.13 RESOLUTION

### Description

The RESOLUTION method is run through when the current resolution has changed for an opened screen, i.e. typically at a TCU switchover.

### Programming

| | |
|---|---|
| Syntax: | RESOLUTION<br><instructions><br>END_RESOLUTION |
| Description: | Resolution |
| Parameters: | - None - |

### See also

Variables S_RESX and S_RESY (Page 89)

## 6.2.14 RESUME

### Description

The RESUME method is called when the screen was interrupted, e.g. for an area change, and is now shown again. The value changes are processed again, if required, the timers started and the RESUME block executed.

### Programming

| | |
|---|---|
| Syntax: | RESUME<br><instruction><br>END_RESUME |
| Description: | Screen active again |
| Parameters: | - None - |

## 6.2.15 SUSPEND

### Description

The SUSPEND method is called when the screen is interrupted and not unloaded. This is the case, for example, when the screen is exited by a simple area change, but not explicitly unloaded. The screen is retained in the background, but no value change or timers are processed. The screen is paused.

### Programming

| | |
|---|---|
| Syntax: | SUSPEND<br><instruction><br>END_SUSPEND |
| Description: | Interruption of the screen |
| Parameters: | - None - |

### Example

```
SUSPEND
   MyVar1 = MyVar1 + 1
END_SUSPEND
```

## 6.2.16 Example: Version management with OUTPUT methods

### Overview

Additional variables can be added to existing dialogs when expanding the user interface. A version identifier in parentheses is appended to the additional variables in the definition following the variable name: (0 = Original, is not written), 1 = Version 1, 2 = Version 2, etc.

**Example:**

```
DEF var100=(R//1)              ; Original, corresponds to Version 0
DEF var101(1)=(S//"Hello")     ; Extension as of Version 1
```

When writing the OUTPUT method, you can specify which variables are written, with reference to a particular version identifier.

**Example:**

```
OUTPUT(NC1)                    ; Only the variables of the original are offered in
                               the OUTPUT method.
```

```
OUTPUT(NC1,1)                    ; The variables of the original and the extensions
                                 with version identifier 1 are offered in the OUTPUT
                                 method.
```

The OUTPUT method for the original does not need a version identifier, however you can specify it with 0. OUTPUT(NC1) is equivalent to OUTPUT(NC1,0). Version identifier n in the OUTPUT method includes all variables of the originals 0, 1, 2, ... up to and including n.

## Programming with version identifier

```
//M(XXX)                         Version 0 (default)
DEF var100=(R//1)
DEF var101=(S//"Hello")
DEF TMP
VS8=("GC")
PRESS(VS8)
   GC("NC1")
END_PRESS

OUTPUT(NC1)
var100",,"var101
END_OUTPUT

; ************ Version 1, extended definition ***************
//M(XXX)
DEF var100=(R//1)
DEF var101=(S//"Hello")
DEF var102(1)=(V//"HUGO")
DEF TMP
VS8=("GC")
PRESS(VS8)
   GC("NC1")
END_PRESS
...

OUTPUT(NC1)                      Original and the new version in addition
var100","var101
END_OUTPUT
...

OUTPUT(NC1,1)                    Version 1
var100","var101"," var102
END_OUTPUT
```

# 6.3 Functions

## Overview

A variety of functions are available in dialogs and dialog-dependent softkey menus. These can be activated by specific events (exit input field, actuate softkey) and configured in methods.

## Subprograms

Repeatedly used configuring instructions or others, which define the process for a particular operation can be configured in subprograms. Subprograms can be loaded into the main program or other subprograms at any time and executed as often as necessary, i.e. the instructions they contain do not need to be configured repeatedly. The definition blocks of the dialogs/softkey menu constitute a main program.

## External functions

Additional, user-specific functions can be integrated by means of external functions. The external functions are stored in a DLL file and identified by an entry in the definition lines of the configuration file.

## PI services

The PI_START function starts PI Services (Program Invocation Services) from the PLC in the NC area.

## See also

Function (FCT) (Page 127)

PI services (Page 141)

## 6.3.1 Reading and writing drive parameters: RDOP, WDOP, MRDOP

## Description

You can read and write drive parameters with the aid of RDOP, WDOP and MRDOP functions.

**Note**

Do not read drive parameters faster than in a 1 second cycle, slower is better.

Reason: The communication with the drives can otherwise be extremely disturbed or even cause failures.

**Note**

If errors occur when reading and writing drive parameters, the screen property ERR is set appropriately.

## Programming

| Syntax: | **RDOP***("Identifier of the drive object", "Parameter number")* | |
|---|---|---|
| Description: | Reading a drive parameter (Drive Object Parameter) | |
| Parameter: | Identifier of the drive object | The identifier of the drive object can be found in the "DO name" column in "Drive system diagnostics" in the diagnostics operating area ( > ETC > HSK 8: Drive system) (see figure below). |
| | Parameter number | |

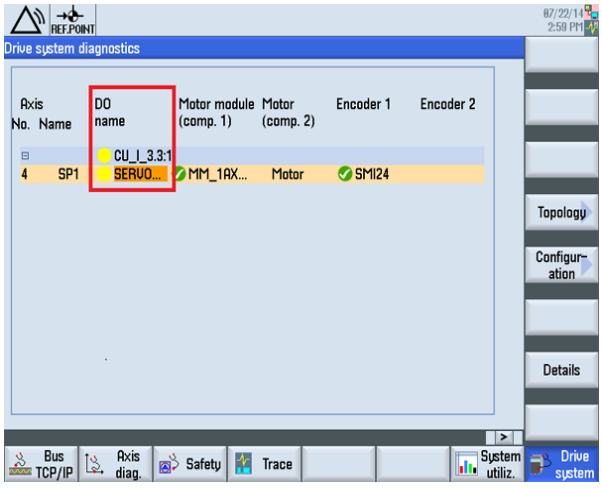



Figure 6-1 Identifier of the drive object

| Syntax: | **MRDOP***("Identifier of the drive object", "Parameter number1"*"Parameter number"[*…], Register index)* |
|---|---|
| Description: | Multi-read of drive parameters.<br>The MRDOP command transfers several drive parameters or drive objects in a single register access. This access method is significantly faster than reading via individual access attempts. |

| | | |
|---|---|---|
| Parameter: | Identifier of the drive object | The identifier of the drive object can be taken, for example, from the root screen of the "Setup" operating area. |
| | Parameter number1 ... n | In the variable names, "*" is the separator. The values are transferred to register REG[Register index] in the order that the variable names appear in the command. |
| | Register index | The value of the first variable is located in REG[Register index].<br>The value of the second variable is located in REG[Register index + 1] |

| | | |
|---|---|---|
| Syntax: | **WDOP***("Identifier of the drive object", "Parameter number", Value)* | |
| Description: | Writing a drive parameter (Drive Object Parameter) | |
| Parameter: | Identifier of the drive object | The identifier of the drive object can be taken, for example, from the root screen of the "Setup" operating area. |
| | Parameter number | Parameter number |
| | Value | Value to be written |

### Examples

Read motor temperature r0035 of drive object "SERVO_3.3:2":

```
MyVar=RDOP("SERVO_3.3:2","35") ;
```

Read motor temperature r0035 and actual torque value r0080 of drive object "SERVO_3.3:2" and store the respective results as of register index 10:

```
MRDOP("SERVO_3.3:2","35*80",10)
```

## 6.3.2 Subprogram call (CALL)

### Description

The CALL function calls a loaded subprogram from any point in a method. Subprogram nesting is supported, i.e. you can call a subprogram from another subprogram.

### Programming

| Syntax: | **CALL***("Identifier")* | |
|---|---|---|
| Description: | Subprogram call | |
| Parameter: | Identifier | Name of subprogram |

### Example

```
//M(SCREEN FORM1)
DEF VAR1 = ...
DEF VAR2 = ...
CHANGE(VAR1)
  ...
  CALL("MY_UP1")              ; Call subprogram and execute
  ...
END_CHANGE

CHANGE(VAR2)
  ...
  CALL("MY_UP1")              ; Call subprogram and execute
  ...
END_CHANGE

SUB(MY_UP1)
                              ;Do something
END_SUB
//END
```

## 6.3.3 Define block (//B)

### Description

In the program file, subprograms are identified by the block identifier //B and terminated with // END. Several subprograms can be defined under each block identifier.

**Note**

The variables used in the subprogram must be defined in the dialog in which the subprogram is called.

### Programming

A block is structured in the following way:

| | | |
|---|---|---|
| Syntax: | **//B**(*Block name*)<br>**SUB***(Identifier)*<br>END_SUB<br>[SUB(*Identifier*)<br>...<br>END_SUB]<br>...<br>//END | |
| Description: | Defines a subprogram | |
| Parameters: | Block name | Name of block identifier |
| | Identifier | Name of subprogram |

### Example

```
//B(PROG1)                        ; Block start
SUB(UP1)                          ; Start of subprogram
  ...
  REG[0] = 5                      ; Assign value 5 to register 0
  ...
END_SUB                           ; End of subprogram
SUB(UP2)                          ; Start of subprogram
  IF VAR1.val=="Otto"
    VAR1.val="Hans"
    RETURN
  ENDIF
  VAR1.val="Otto"
END_SUB                           ; End of subprogram
//END                             ; Block end
```

## 6.3.4 Check Variable (CVAR)

### Description

You can use the CVAR (CheckVariable) function to run a scan to ascertain whether all or only certain variables or Help variables in a screen form are error-free.

It may be useful to check if variables contain a valid value before an NC code with the GC function.

A variable is error-free if the state of the variable Identifier.vld = 1.

### Programming

| Syntax: | **CVAR**(*VarN*) | |
|---|---|---|
| Description: | Checks variables for valid content | |
| Parameters: | VarN | List of variables to be checked.<br>Up to 29 variables, each separated by a comma, can be checked. A character length of 500 must not be exceeded.<br>The result of the scan can be: |
| | 1 = | TRUE (all variables have valid content) |
| | 0 = | FALSE (at least one variable has invalid content) |

### Example

```
IF CVAR == TRUE                          ; Check all variables
  VS8.SE = 1                             ; If all variables are error-free, soft-
                                         key VS8 is visible
ELSE
  VS8.SE = 2                             ; If a variable has an invalid value,
                                         softkey VS8 is disabled
ENDIF

IF CVAR(”VAR1”, ”VAR2”) == TRUE
                                         ; Check variables VAR1 and VAR2
  DLGL ("VAR1 and VAR2 are OK")
                                         ; If the values of VAR1 and VAR2 are er-
                                         ror-free, "VAR1 and VAR2 are OK" appears
                                         in the dialog line
ELSE
  DLGL ("VAR1 and VAR2 are not OK")
                                         ; If the values of VAR1 and VAR2 are in-
                                         valid, "VAR1 and VAR2 are not OK" appears
                                         in the dialog line
ENDIF
```

## 6.3.5 CLEAR_BACKGROUND

### Description

The CLEAR_BACKGROUND function deletes the LINE, RECT, ELLIPSE, V_SEPARATOR and H_SEPARATOR graphic elements.

### See also

Line, dividing line, rectangle, circle and ellipse (Page 163)

## 6.3.6 Copy Program file function (CP)

### Description

The CP (Copy Program) function copies files within the HMI file system or within the NC file system.

### Programming

| Syntax: | **CP***("Source file", "Target file")* | |
|---|---|---|
| Description: | Copies a file | |
| Parameter: | Source file | Complete path to the source file |
| | Target file | Complete path data of the target file |

The return value (VAR1 defined as help variable) queries whether the function was successful:

```
CP("//NC/MPF.DIR/HOHO.MPF","//NC/MPF.DIR/ASLAN.MPF",VAR1)
```

### Example

Application with return value:

```
CP("//NC/MPF.DIR/HOHO.MPF","//NC/MPF.DIR/ASLAN.MPF",VAR1)
CP("CF_CARD:/wks.dir/MESS_BILD.WPD/MESS_BILD.MPF","//NC/WKS.DIR/AAA.WPD/HOHO2.MPF",VAR1)
CP("//NC/MPF.DIR/HOHO.MPF","//NC/MPF.DIR/ASLAN.MPF")                    ; WPD must exist
```

Application without return value:

```
CP("//NC/MPF.DIR/HOHO.MPF","//NC/MPF.DIR/ASLAN.MPF")
CP("CF_CARD:/mpf.dir/MYPROG.MPF","//NC/MPF.DIR/HOHO.MPF")
CP("//NC/MPF.DIR/HOHO.MPF","CF_CARD:/XYZ/MYPROG.MPF")                   ; XYZ must exist
```

### See also

Support of FILE_ERR: FILE_ERR variable (Page 84)

## 6.3.7 Delete Program file function (DP)

### Description

The DP (Delete Program) function deletes a file from the passive HMI or active NC file system.

### Programming

| | | |
|---|---|---|
| Syntax: | **DP***("File")* | |
| Description: | Delete file | |
| Parameter: | File | Complete path name of file to be deleted |

### Example

The following data management syntax is used for this function:

- With return value

  ```
  DP("//NC/MPF.DIR/MYPROG.MPF",VAR1)
  DP("//NC/WKS.DIR/TEST.WPD/MYPROG.MPF",VAR1)
  DP("//NC/CMA.DIR/MYPROG.SPF",VAR1)
  ```

  VAR1 = 0 File was deleted.

  VAR1 = 1 File was not deleted.
- Without return value:

  ```
  DP("//NC/MPF.DIR/MYPROG.MPF")
  DP("//NC/WKS.DIR/TEST.WPD/MYPROG.MPF")
  DP("//NC/CMA.DIR/MYPROG.SPF")
  ```

## 6.3.8 Exist Program file function (EP)

### Description

The EP (Exist Program) function checks whether a particular NC program is stored on the specified path in the NC or HMI file system.

### Programming

| | | |
|---|---|---|
| Syntax: | **EP***("File")* | |
| Description: | Checks the existence of the NC program | |
| Parameter: | File | Complete path to the file in the NC or HMI file system |
| Return value: | Name of a variable to which the result of the scan should be assigned | |

The EP function can handle the new syntax and the old logic (with adapted syntax).

The file is directly addressed using a qualifying name:

```
//NC/MPF.DIR/XYZ.MPF
```

or
`CF_CARD: /MPF.DIR/XYZ.MPF` (points to /user/sinumerik/data/prog)
or
`LOC:` (corresponds to CF_CARD)

**Examples of new syntax:**

```
EP("//NC/WKS.DIR/TEST.WPD/XYZ.MPF",VAR1)
EP("CF_CARD:/mpf.dir/XYZ.MPF",VAR1)
EP("LOC:/mpf.dir/XYZ.MPF",VAR1)
; With return value:
; VAR1 = 0 File exists.
; VAR1 = 1 File does not exist.
```

**Example of old syntax:**

```
EP("/MPF.DIR/CFI.MPF", VAR1)
; With return value:
; VAR1 = M File is located in the HMI file system.
; VAR1 = N File is located in the NC file system.
; VAR1 = B File is located in the HMI and the NC file system.
```

## Example

```
EP("/MPF.DIR/GROB.MPF",VAR1)                                    ; Old - path now with / instead of \

IF VAR1 == "M"
   DLGL("File is located in the HMI file system")
ELSE
   IF VAR1 == "N"
      DLGL("File is located in the NC file directory")
   ELSE
     DLGL("File is located neither in the HMI nor in the NC
file system")
   ENDIF
ENDIF
```

## 6.3.9 Move Program file function (MP)

## Description

The MP (Move Program) function copies files within the HMI file system or within the NC file system.

### Programming

| Syntax: | **MP***("source", "target")* | |
|---|---|---|
| | `MP("CF_CARD:/MPF.DIR/MYPROG.MPF","//NC/MPF.DIR")` | |
| Description: | Move file | |
| Parameter: | Source file | Complete path data |
| | Target file | Complete path data |
| Return value | Result of the query | |

### Examples

With return value:

```
MP("//NC/MPF.DIR/123.MPF","//NC/MPF.DIR/ASLAN.MPF",VAR1)          ;Within NC
MP("//NC/MPF.DIR/123.MPF",VAR0,VAR1)                              ;Target via variable
MP(VAR4,VAR0,VAR1)                                                ;Source and target via variable
MP("CF_CARD:/mpf.dir/myprog.mpf","//NC/MPF.DIR/123.MPF",VAR1)     ;From CF card to NC
MP("//NC/MPF.DIR/HOHO.MPF","CF_CARD:/XYZ/123.mpf",VAR1)           ;From NC to CF card

; With return value:
; VAR1 = 0 Executed
; VAR1 = 1 Not executed
```

## 6.3.10 Select Program file function (SP)

### Description

The SP (Select Program) function selects a file in the active NC file system for execution, i.e. the file must be loaded into the NC beforehand.

### Programming

| Syntax: | **SP***("File")* | |
|---|---|---|
| Designation: | Select program | |
| Parameter: | "File" | Complete path name of NC file |

### Example

The following data management syntax is used for this function:

- With return value

  `SP("//NC/MPF.DIR/MYPROG.MPF", VAR1)`

  VAR1 = 0 File was loaded.

VAR1 = 1 File was not loaded without return value.

- Without return value:

```
SP("//NC/MPF.DIR/MYPROG.MPF")
```

## 6.3.11 File accesses: RDFILE, WRFILE, RDLINEFILE, WRLINEFILE

### Description

The RDFILE and WRFILE functions are available for read and write access to files with INI syntax.

The RDLINEFILE and WRLINEFILE functions are available for read and write access to individual lines of a file.

### Programming

| Syntax: | **RDFILE**(*"File name + path", "Section", "Key"*) | |
|---|---|---|
| Description: | Reading from a file | |
| Parameter: | File name + path | Path and file name |
| | Section | Section in the INI file |
| | Key | Key in the INI file |

| Syntax: | **WRFILE***(Value, "File name + path", "Section", "Key")* | |
|---|---|---|
| Description: | Writing to a file | |
| Parameter: | Value | Value to be written |
| | File name + path | Path and file name |
| | Section | Section in the INI file |
| | Key | Key in the INI file |

| Syntax: | **RDLINEFILE***("File name + path", Line number)* | |
|---|---|---|
| Description: | Reading a line from a file | |
| Parameter: | File name + path | Path and file name |
| | Line number | Line number<br>Numbering begins at 0 for the first line. |

| Syntax: | **WRLINEFILE***(Value, "File name + path")* |
|---|---|
| Description: | Writing of a line at the end of a file |

| Parameter: | Value | Value to be written |
|---|---|---|
| | File name + path | Path and file name |

**Note**

- The files must not be in the file system of the NC (data management).
- If the file does not exist, or the end of the file is reached or other errors occur, the variables FILE_ERR and ERR are set accordingly. You can check whether a file exists first with the Exist Program (EP) file function.
- The processed file is generated in "UTF-8" coding (without BOM (Byte Order Mask). When reading a file, it is expected in "UTF-8" coding.
- You can delete the file explicitly when required with the Delete Program (DP) file function.

## Examples

**Reading from an INI file:**

Requirement/assumption:

Content of file C:/tmp/myfile.ini:

<…>
```
[MyData]
MyName=Daniel
```
<…>

```
MyVar = RDFILE("C:/tmp/myfile.ini", "MyData", "MyName")
```

Result:

MyVar now contains the value "Daniel".

**Writing to an INI file:**

Requirement/assumption:

VARS=12

```
WRFILE(VARS, "C:/tmp/myfile.ini", "MySession", "NrOfSessions")
```

Result:

Content of file C:/tmp/myfile.ini:

<…>
```
[MySession]
NrOfSessions=12
```
<…>

**Reading the 4th line of a file:**

Requirement/assumption:

Content of c:/tmp/myfile.mpf:

```
R[0]=0
F3500 G1
MYLOOPIDX1: X0 y-50 Z0
  X150 Y50 Z10
  R[0]=R[0]+1
GOTOB MYLOOPIDX1
M30
```

```
MyVar = RDLINEFILE("C:/tmp/myfile.mpf", 4)
```

Result:

MyVar now contains the value "  R[0]=R[0]+1"

**Writing at the end of a file:**

Requirement/assumption:

```
VARX=123
VARY=456
```

```
WRLINEFILE("F100 X" << VARX << " Y" << VARY, "C:/tmp/mypp.mpf")
```

Result:

Content of c:/tmp/mypp.mpf:

```
<...>
F100 X123 Y456
```

## 6.3.12 Dialog line (DLGL)

### Description

It is possible to configure short texts (messages or input tips) for output in the dialog line of the dialog in response to certain situations.

Possible number of characters in the default font size: approx. 50

### Programming

| Syntax: | **DLGL***("String")* | |
|---|---|---|
| Description: | Outputs text in the dialog line | |
| Parameter: | String | Text, which is displayed in the dialog line |

### Example

```
IF Var1 > Var2
                              ; The text "Value too large!" appears in the dialog
                              line if variable1 > variable2.
  DLGL("Value too large!")
ENDIF
```

## 6.3.13 DEBUG

### Description

The DEBUG function provides an analysis help during the draft phase of Run MyScreen user screens. An expression in brackets transferred with the DEBUG function is evaluated during runtime. The result is attached to the "easyscreen_log.txt" logbook as a separate entry.

Each entry has the current time stamp in square brackets as prefix (see example below).

It is recommended that the DEBUG outputs are removed in time-critical parts when possible. Particularly after completing the draft phase, it is recommended that you comment out the DEBUG outputs. The write access to the constantly expanding "easyscreen_log.txt" logbook may slow down the processing.

### Programming

| Syntax: | **DEBUG***(expression)*) |
|---|---|
| Description: | Makes an entry in the "easyscreen_log.txt" logbook |
| Parameter: | Expression to be evaluated from which an entry is generated in the logbook |

### Example

```
IF Var1 > Var2
  DEBUG("Value of ""Var1"": " << Var1)
                                        ; Entry in the "easyscreen_log_txt:
                                        [10:22:40.445] DEBUG: Value of "Var1":
                                        123.456
ENDIF
```

## 6.3.14 Exit dialog (EXIT)

### Description

The EXIT function is used to exit a dialog and return to the master dialog. If no master dialog is found, you will exit the newly configured user interfaces and return to the standard application.

### Programming (without parameters)

| | |
|---|---|
| Syntax: | **EXIT** |
| Description: | Exits a dialog |
| Parameters: | - None - |

### Example

```
PRESS(HS1)
  EXIT
END_PRESS
```

### Description

If the current dialog has been called with a transfer variable, the value of the variables can be changed and transferred to the output dialog.

The variable values are each assigned to the variables transferred from the output dialog to the subsequent dialog using the "LM" function. Up to 20 variable values, each separated by a comma, can be transferred.

**Note**

The sequence of variables or variable values must be the same as the sequence of transfer values programmed for the LM function to preclude assignment errors. Any unspecified variable values will not be changed when the transfer is made. The modified transfer variables are immediately valid in the output dialog on execution of the LM function.

### Programming with a transfer variable

| | | |
|---|---|---|
| Syntax: | **EXIT***[(VARx)]* | |
| Description: | Exits the dialog and transfers one or more variables | |
| Parameters: | VARx | Label variables |

### Example

```
//M(Screen form1)
...
PRESS(HS1)
  LM("SCREEN FORM2","CFI.COM",1, POSX,
POSY, DIAMETER)
                                        ; Interrupt screen form1 and open screen
                                        form2. In doing this, transfer variables
                                        POSX, POSY and DIAMETER.
  DLGL("Screen form2 ended")            ; After returning from screen form2, the
                                        following text appears in the dialog line
                                        of screen form1: Screen form2 ended.
END_PRESS
...
//END

//M(Screen form2)
...
PRESS(HS1)
EXIT(5, , CALCULATED_DIAMETER)
                                        ; Exit screen form2 and return to screen
                                        form1 in the line after LM. In doing
                                        this, assign the value 5 to the variable
                                        POSX and the value of the CALCULATED_DI-
                                        AMETER variable to the DIAMETER variable.
                                        The variable POSY retains its current val-
                                        ue.
END_PRESS
...
//END
```

## 6.3.15 Dynamic manipulation of the lists of toggle fields or list box fields

### Description

The LISTADDITEM, LISTINSERTITEM, LISTDELETEITEM and LISTCLEAR functions are used for the dynamic manipulation of the lists of toggle fields or list box fields.

These functions only have an effect on variables that have their own list, such as:

- "Simple" list
  `DEF VAR_AC1 = (I/* 0,1,2,3,4,5,6,7,8)` or
- "Extended" list
  `DEF VAR_AC2 = (I/* 0="AC0", 1="AC1", 2="AC2", 3="AC3", 4="AC4", 5="AC5", 6="AC6", 7="AC7", 8="AC8").`

If the variable points to an array, e.g. `DEF VAR_AC3 = (I/* MYARRAY)`, these functions are not available as otherwise the global array would be changed.

At least one value must be defined for the variable in the DEF line. This specifies the "Simple" or "Extended" list type.

However, the list can then be completely deleted and, if required, completely rebuilt. The "Simple" or "Extended" type must be retained however, it cannot be changed dynamically.

## Programming

| Syntax: | **LISTINSERTITEM***(Variable name, Position, ItemValue[, ItemDisp-Value])* | |
|---|---|---|
| Description: | Inserts an element at a specific position; reading from a file | |
| Parameter: | Variable name | |
| | Position | Position at which an element is to be inserted in the list |
| | ItemValue | Value of the list entry |
| | ItemDispValue | Value as it is to be displayed in the list |

| Syntax: | **LISTADDITEM**(Variable name, ItemValue[, ItemDispValue]) | |
|---|---|---|
| Description: | Adds an element to the end of the list | |
| Parameter: | Variable name | |
| | ItemValue | Value of the list entry |
| | ItemDispValue | Value as it is to be displayed in the list |

| Syntax: | **LISTDELETEITEM**(Variable name, Position) | |
|---|---|---|
| Description: | Deletes a specific element | |
| Parameter: | Variable name | |
| | Position | Position of the element to be deleted in the list |

| Syntax: | **LISTCOUNT**(Variable name) | |
|---|---|---|
| Description: | Returns the current number of elements in the list | |
| Parameter: | Variable name | |

| Syntax: | **LISTCLEAR***(Variable name)* | |
|---|---|---|
| Description: | Deletes the complete list | |
| Parameter: | Variable name | |

## Examples

**Note**

The following examples build upon each other. The sequence of the examples is relevant in order to understand the respective results.

Requirement/assumption:

```
DEF VAR_AC = (I/* 0="Off",1="On"/1/,"Switch"/WR2)
```

**Adding an element -1="Undefined" to the end of the list:**

```
LISTADDITEM("VAR_AC", -1, """Undefined""")
```

Result: `0="Off", 1="On", -1="Undefined"`

**Inserting an element 99="Maybe" at position 2:**

```
LISTINSERTITEM("VAR_AC", 2, 99, """Maybe""")
```

Result: `0="Off", 1="On", 99="Maybe", -1="Undefined"`

**Determining the current number of list elements:**

```
REG[10]=LISTCOUNT("VAR_AC")
```

Result: REG[10] = 4

**Deleting the element at position 1:**

```
LISTDELETEITEM("VAR_AC", 1)
```

Result: `0="Off", 99="Maybe", -1="Undefined"`

**Deletion of the complete list:**

```
LISTCLEAR("VAR_AC")
```

Result: List is empty

## 6.3.16 Evaluate (EVAL)

### Description

The EVAL function evaluates a transferred expression and then executes it. With this function, expressions can be programmed during runtime. This can be useful, for example, for indexed access operations to variables.

### Programming

| Syntax: | **EVAL***(exp)* | |
|---|---|---|
| Description: | Evaluates an expression | |
| Parameters: | exp | Logic expression |

### Example

```
VAR1=(S)
VAR2=(S)
VAR3=(S)
VAR4=(S)
CHANGE()
  REG[7] = EVAL("VAR"<<REG[5])  ; The expression in parentheses produces VAR3 if the
                                value of REG[5] is equal to 3. The value of VAR3 is,
                                therefore, assigned to REG[7].
  IF REG[5] == 1
    REG[7] = VAR1
  ELSE
    IF REG[5] == 2
      REG[7] = VAR2
    ELSE
      IF REG[5] == 3
        REG[7] = VAR3
      ELSE
        IF REG[5] == 4
          REG[7] = VAR4
        ENDIF
      ENDIF
    ENDIF
  ENDIF
END_CHANGE
```

## 6.3.17 Exit Loading Softkey (EXITLS)

### Description

You can use the EXITLS function to exit the current user interface and load a defined softkey menu.

### Programming

| Syntax: | **EXITLS***("Softkey menu"[, "Path*"]) | |
|---|---|---|
| Description: | Exits dialog and loads a softkey menu | |
| Parameters: | Softkey menu | Name of the softkey menu to be loaded |
| | Path name | Directory path of the softkey menu to be loaded |

### Example

```
PRESS(HS1)
    EXITLS( "Menu1", "AEDITOR.COM" )
END_PRESS
```

## 6.3.18 Function (FCT)

### Description

The external functions are stored in a DLL file and identified by an entry in the definition lines of the configuration file.

**Note**

The external function must have at least one return parameter.

### Programming

| Syntax: | **FCT***Function name = ("File"/Type of return/Types of permanent parameters/ Types of variable parameters)* |
|---|---|
| | FCT InitConnection = ("c:\tmp\xyz.dll"/I/R,I,S/I,S) |
| Description: | An external function can, for example, be called from the LOAD method or executed in the PRESS method. |

| | | |
|---|---|---|
| Parameters: | Function name | Name of external function |
| | File | Complete path to DLL file |
| | Type of return | Data type of the return value |
| | Type of fixed parameter | Value parameter |
| | Type of variable parameter | Reference parameter |
| | The data types are separated by commas. | |

The external function can, for example, be called from the LOAD method or executed in the PRESS method.

**Example:**

```
press(vs4)
RET = InitConnection(VAR1,13,"Servus",VAR2,VAR17)
end_press
```

## Structure of the external function

The external function must take into account a certain, specific signature:

| | | |
|---|---|---|
| Syntax: | extern "C" dllexport void InitConnection (ExtFctStructPtr FctRet, ExtFctStructPtr FctPar, char cNrFctPar) | |
| Description: | DLL export, only when implemented in Windows<br>Specifiers and transfer parameters are strictly defined. The actual call parameters are transferred using the transferred structures. | |
| Parameters: | cNrFctPar | Number of call parameters = number of structure elements in FctPar |
| | FctPar | Pointer to a field of structure elements, which contain the particular call parameter with data type. |
| | FctRet | Pointer to a structure for the function value return with data type. |

## Definition of the transfer structure

```
union CFI_VARIANT
    (
    char                                b;
    short int                           i;
    double                              r;
    char*                               s;
    )
typedef struct ExtFctStructTag
    (
```

```
    char                                cTyp;
    union CFI_VARIANT                   value;
    )ExtFctStruct;
typedef struct ExtFct* ExtFctStructPtr;
```

If the external function is to be developed independently of the platform (Windows, Linux), then it is not permissible to use the keyword __declspec(dllexport). This keyword is only required under Windows. For instance, the following macro can be used under Qt:

```
#ifdef Q_WS_WIN
       #define MY_EXPORT __declspec(dllexport)
#else
       #define MY_EXPORT
#endif
```

The function is declared as follows:

```
extern "C" MY_EXPORT void InitConnection
(ExtFctStructPtr FctRet, ExtFctStructPtr FctPar,char cNrFctPar)
```

If the screens, configured with "Run MyScreens", are used on the NCU and PCU/PC, then the extension of the binary file must be omitted:

```
FCT InitConnection = ("xyz"/I/R,I,S/I,S)
```

When the absolute path information is omitted, "Run MyScreens" first searches for the binary file in the configured directory.

## 6.3.19 Generate code (GC)

### Description

The GC (Generate Code) function generates NC code from the OUTPUT method.

### Programming

| | |
|---|---|
| Syntax: | **GC**("*Identifier"[,"Target file"][,Opt],[Append])* |
| Description: | Generates NC code |

| | | |
|---|---|---|
| Parameter: | Identifier | Name of the OUTPUT method from which code is generated |
| | Target file | Path name of target file for HMI or NC file system<br>If the target file is not specified (only possible within Programming Support), the code will be written to the location of the cursor within the file that is currently open. |
| | Opt | Option for generating comments. |
| | 0: | (Default setting) Generate code with comment for the purpose of recompilability (see also Recompile (Page 146)). |
| | 1: | Do not create comments in the generated code.<br>**Note:** This code cannot be recompiled (see also Auto-Hotspot). |
| | Append | This parameter is only relevant if a target file is specified. |
| | 0: | (Default setting) If the file already exists, the old content is deleted. |
| | 1: | If the file already exists, the new code is written at the start of the file. |
| | 2: | If the file already exists, the new code is written at the end of the file. |

## Example

```
//M(TestGC/"Code generation:")
DEF VAR1 = (R//1)
DEF VAR2 = (R//2)
DEF D_NAME
LOAD
  VAR1 = 123
  VAR2 = -6
END_LOAD
OUTPUT(CODE1)
  "Cycle123(" VAR1 "," VAR2 ")"
  "M30"
END_OUTPUT

PRESS(VS1)
  D_NAME = "\MPF.DIR\MESSEN.MPF"
  GC("CODE1",D_NAME)                ; Write NC code from the OUTPUT method to file
                                    \MPF.DIR\MESSEN.MPF:
                                    Cycle123(123, -6)
                                    M30
END_PRESS
```

## Recompile

- **No entry for target file:**
  The GC function can only be used in the Programming Support and writes the NC code to the file currently open in the editor. Recompilation of the NC code is possible. If the GC function is configured without a target file being specified under "Run MyScreens", an error message is output when it is executed.
- **Entry for target file:**
  The code generated from the OUTPUT method is entered in the target file. If the target file does not already exist, it is set up in the NC file system. If the target file is stored in the HMI file system, it is stored on the hard disk. User comment lines (information required to recompile code) are not set up, i.e. the code cannot be recompiled.

## Special considerations for target file specification

In principle, there are two different ways of specifying a target file:

- **NC notation:** `/_N_MPF_DIR/_N_MY_FILE_MPF`
  The file is created in the MPF directory on the NC.
- **DOS notation:** `d:\abc\my_file.txt` or `\\RemoteRechner\files\my_file.txt`
  The file is written to the specified directory on the hard disk or on the specified PC, provided that the directory is available on the hard disk or on a remote PC.

**Note**

Invalid variables generate an empty string in generated NC code and an error message in the log book when they are read.

## Special features of recompilation

The GC function cannot be called in sub-dialogs because variables originating from master dialogs can be used in sub-dialogs. These variables would not, however, be available in response to a direct call.

When generated code is processed manually with the editor, the number of characters for values created by the code generation program must not be changed. Changing these values would make it impossible to recompile the code.

Remedy:

1. Recompile
2. Make change using the configured dialog (e.g. 99 → 101)
3. GC

## 6.3.20 Password functions

### Overview

The following password functions are available:

- Set password
- Delete password
- Change password

### HMI_LOGIN function: Set new password

| Syntax: | **HMI_LOGIN***(Passwd)* | |
|---|---|---|
| Description: | A password which sets the current access level is sent to the NCK with the HMI_LOGIN() function. | |
| Parameters: | Passwd | Password |

### Example

```
REG[0] = HMI_LOGIN("CUSTOMER")          ; Result = TRUE if the password has been set
                                          successfully, otherwise FALSE
```

### HMI_LOGOFF function: Delete password

| Syntax: | **HMI_LOGOFF** |
|---|---|
| Description: | The current access level can be reset with the HMI_LOGOFF function. |
| Parameters: | - None - |

### Example

```
REG[0] = HMI_LOGOFF                     ; Result = TRUE if the password has been de-
                                          leted successfully, otherwise FALSE
```

### HMI_SETPASSWD function: Change password

| Syntax: | **HMI_SETPASSWD***(AC, Passwd)* | |
|---|---|---|
| Description: | The password of the current level or a lower level can be changed with the HMI_SETPASSWD function. | |
| Parameters: | AC | Access level |
| | Passwd | Password |

### Example

```
REG[0] = HMI_SETPASSWD(4,"MYPWD")      ; Result = TRUE if the password has been
                                       changed successfully, otherwise FALSE
```

## 6.3.21 Load Array (LA)

### Description

The LA (Load Array) function can be used to load an array from another file.

### Programming

| Syntax: | **LA***(Identifier [, File])* | |
|---|---|---|
| Description: | Loads an array from a file | |
| Parameters: | Identifier | Name of array to be loaded |
| | File | File in which the array is defined |

**Note**

If an array in the current configuration file must be replaced by an array from another configuration file, then both arrays must have the same name.

### Example

```
                                       ; Extract from file  maske.com
DEF VAR2 = (S/*ARR5/"Out"/,"Toggle field")
PRESS(HS5)
  LA("ARR5","arrayext.com")            ; Load array ARR5 from file arrayext.com
  VAR2 = ARR5[0]                       ; "Above"/"Below"/"Right"/"Left" appears
                                       in the VAR2 toggle field
                                       instead of "Out/In"
END_PRESS
//A(ARR5)
("Out"/"In")
//END
                                       ; Extract from file arrayext.com
//A(ARR5)
("Above"/"Below"/"Right"/"Left")
//END
```

**Note**

Please note that a valid value must be assigned to a variable after the LA function has been used to assign another array to the toggle field of the variable.

## 6.3.22 Load Block (LB)

### Description

The LB (Load Block) function can be used to load blocks containing subprograms during runtime. LB should be configured in a LOAD method so that the loaded subprograms can be called at any time.

**Note**

Subprograms can also be defined directly in a dialog so that they do not have to be loaded.

### Programming

| Syntax: | **LB***("Block name"[,"File"])* | |
|---|---|---|
| Description: | Loads subprogram during runtime | |
| Parameters: | Block name | Name of block identifier |
| | File | Path name of configuration file<br>Default setting = Current configuration file |

### Example

```
LOAD
  LB("PROG1")              ; Block "PROG1" is searched for in the current configura-
                           tion file and then loaded.
  LB("PROG2","XY.COM")     ; Block "PROG2" is searched for in the configuration file
                           XY.COM and then loaded.
END_LOAD
```

## 6.3.23 Load Mask (LM)

### Description

The LM function loads a new dialog. Depending on the mode of the dialog change (see table below), a message dialog can also be implemented with this function.

### Master dialog / sub-dialog

A dialog, which calls another dialog, but is not ended itself, is referred to as a master dialog. A dialog that is called by a master dialog is referred to as a sub-dialog.

### Programming

| | | |
|---|---|---|
| Syntax: | **LM***("Identifier"[,"File"] [,MSx [, VARx] ] )* | |
| Description: | Loads a dialog | |
| Parameter: | Identifier | Name of the dialog to be loaded |
| | File | Path name (HMI file system or NC file system) of the configuration file; default setting: Current configuration file |
| | MSx | Mode of dialog change |
| | 0: | (Default setting) The current dialog is rejected, the new dialog is loaded and displayed. EXIT will send you back to the standard application. You can use the MSx parameter to determine whether or not the current dialog should be terminated when changing dialogs. If the current dialog is retained, variables can be transferred to the new dialog.<br>The advantage of the MSx parameter is that the dialogs do not always need to be reinitialized when they are changed. Instead, the data and layout of the current dialog are retained and data transfer is made easier. |
| | 1: | The current master dialog is interrupted when the LM function is initiated. The new sub-dialog is loaded and displayed (e.g. for the implementation of a message dialog). EXIT will end the sub-dialog and return to the point at which the master dialog was interrupted.<br>In the master dialog, the UNLOAD block is not processed during the interruption. |
| | VARx | Precondition: MS1<br>List of variables which can be transferred from the master dialog to the sub-dialog. Up to 20 variables, each separated by a comma, can be transferred. |

**Note**

Parameter VARx transfers only the value of the variable in each case, i.e. variables can be read and written in the sub-dialog, but are not visible in it. Variables can be returned from the sub-dialog to the master dialog by means of the EXIT function.

### Example

**SCREEN FORM1 (master dialog): Jump to the sub-dialog SCREEN FORM2 with variable transfer**

```
PRESS(HS1)

  LM("SCREEN FORM2","CFI.COM",1, POSX, POSY, DIAMETER)
                                    ; Interrupt screen form1 and open screen form2:
                                    In doing this, variables POSX, POSY and DIAME-
                                    TER are transferred.
  DLGL("Screen form2 ended")        ; After returning from screen form2, the fol-
                                    lowing text appears in the dialog line of
                                    screen form1: Screen form2 ended.
END_PRESS
```

**SCREEN FORM2 (sub-dialog): Return to master dialog SCREEN FORM1 with transfer of the variable contents**

```
PRESS(VS8)

  EXIT(POSX, POSY, DIAMETER)
                                    ; Hide screen form2 and continue screen form1:
                                    In doing this, the changed variables POSX, POSY
                                    and DIAMETER are transferred.
END_PRESS
```

## 6.3.24 Load Softkey (LS)

### Description

The LS function can be used to display another softkey menu.

### Programming

| | |
|---|---|
| Syntax: | **LS***("Identifier"[, "File"][, Merge])* |
| Description: | Displays the softkey menu |

| | | |
|---|---|---|
| Parameters: | Identifier | Name of the softkey menu |
| | File | Path (HMI file system or NC file system) to the configuration file<br>Default setting: Current configuration file |
| | Merge | |
| | 0: | All existing softkeys are deleted; the newly configured softkeys are entered. |
| | 1: | Default setting<br>Only the newly configured softkeys overwrite the available softkeys. The other softkeys (= softkeys of the HMI application) are kept with their functionality and text. |

## Example

```
PRESS(HS4)
  LS("Menu2",,0)      ; Menu2 overwrites the existing softkey menu, the softkeys
                      that are displayed are deleted.
END_PRESS
```

**Note**

As long as the Interpreter has not displayed a dialog, i.e. no LM function has yet been processed, only one LS or one LM command, but no other action, can be configured in the PRESS method of the definition block for the start softkey and the softkey menu.

The LS and LM functions may only be called within a softkey PRESS method and will not react if navigation keys are pressed (PU, PD, SL, SR, SU, SD).

## 6.3.25 Load Grid (LG)

### Description

The table description (grid) can be provided dynamically within the LOAD method using the LG method.

In order to assign a table using the LG method, the variable must have already been defined as a grid variable and cross-referenced to an existing, valid table.

### Programming

| | |
|---|---|
| Syntax: | **LG** *(Grid name, Variable name [,File name])* |
| Description: | Loads a table |

| | | |
|---|---|---|
| Parameters: | Grid name | Name of the table (grid) in inverted commas |
| | Variable name | Name of the variable to which the table is to be assigned, in inverted commas |
| | File name | Name of the file in which the table (grid) is defined, in inverted commas. Only needs to be specified if the table is not defined within the file that also contains the definition of the variable. |

### Example

```
//M(MyGridSample/"My Grid Sample")
DEF MyGridVar=(R/% MyGrid1///WR2////100,,351,100)
LOAD
   LG("MyGrid1","MyGridVar","mygrids.com")
END_LOAD
```

**Content of mygrids.com:**

```
//G(MyGrid1/0/5)
(I///,"MyGrid1"/wr1//"1"/80/1)
(R3///"LongText1","R1-R4"/wr2//"$R[1]"/80/1)
(IBB///"LongText2","M2.2-M2.5"/wr2//"M2.2"/80/,1)
(R3///"LongText3","R9,R11,R13,R15"/wr2//"$R[9]"/110/2)
//END
```

## 6.3.26 Multiple selection SWITCH

### Description

You can check a variable for different values with a SWITCH command.

The expression configured in the SWITCH statement is compared in succession to all the values configured in the following CASE statements.

If not the same, a comparison is made with the following CASE statement. If a DEFAULT statement follows and up to now no CASE statement was the same as the expression configured in the SWITCH statement, the statements following the DEFAULT statement are executed.

If the same, the following statements are executed until a CASE, DEFAULT or END_SWITCH statement occurs.

### Example

```
FOCUS
   SWITCH (FOC)
      CASE "VarF"
         DLGL("Variable ""VarF"" has the input focus.")
      CASE "VarZ"
         DLGL("Variable ""VarZ"" has the input focus.")
      DEFAULT
         DLGL("Any other variable has the input focus.")
   END_SWITCH
END_FOCUS
```

## 6.3.27 Multiple Read NC PLC (MRNP)

### Description

This MRNP command transfers several NC/PLC variables in a single register access. This access method is significantly faster than reading via individual access attempts. The NC/PLC variables must be included within an MRNP command of the same area.

The areas of the NC/PLC variables are organized as follows:

- General NC data ($MN..., $SN.., /nck/...)
- Channel-specific NC data ($MC..., $SC.., /channel/...)
- PLC data (DB..., MB.., /plc/...)
- Axis-specific NC data on the same axis ($MA..., $SA..)

### Programming

| | |
|---|---|
| Syntax: | **MRNP***(Variable name 1*Variable name 2[* ...], Register index)* |
| Description: | Reads several variables |
| Parameter: | In the variable names, "*" is the separator. The values are transferred to register REG[Register index] and those following in the order that the variable names appear in the command.<br>The following therefore applies:<br>The value of the first variable is located in REG[Register index].<br>The value of the second variable is located in REG[Register index + 1], etc. |

**Note**

It should be noted that the number of registers is restricted and the list of variables cannot exceed 500 characters.

## Example

```
MRNP("$R[0]*$R[1]*$R[2]*$R[3]",1)      ;The values of variables $R[0] to $R[3] are
                                        written to REG[1] to REG[4].
```

## NC variable

All machine data, setting data and R parameters are available, but only certain system variables (see also: List of accessible system variables (Page 204)).

All global and channel-specific user variables (GUDs) can be accessed. However, local and program-global user variables cannot be processed.

| Machine data | |
|---|---|
| Global machine data | $MN_... |
| Axis-specific machine data | $MA_... |
| Channel-specific machine data | $MC_... |

| Setting data | |
|---|---|
| Global setting data | $SN_... |
| Axis-specific setting data | $SA_... |
| Channel-specific setting data | $SC_... |

| System variables | |
|---|---|
| R parameter 1 | $R[1] |

## PLC variable

All PLC data are available.

| PLC data | |
|---|---|
| Byte y bit z of data block x | DBx.DBXy.z |
| Byte y of data block x | DBx.DBBy |
| Word y of data block x | DBx.DBWy |
| Double word y v. of data block x | DBx.DBDy |
| Real y of data block x | DBx.DBRy |
| Flag byte x bit y | Mx.y |

| PLC data | |
|---|---|
| Flag byte x | MBx |
| Flag word x | MWx |
| Flag double word x | MDx |
| Input byte x bit y | Ix.y or Ex.y |
| Input byte x | IBx or EBx |
| Input word x | IWx or EWx |
| Input double word x | IDx or EDx |
| Output byte x bit y | Qx.y or Ax.y |
| Output byte x | QBx or ABx |
| Output word x | QWx or AWx |
| Output double word x | QDx or ADx |
| String y with length z from data block x | DBx.DBSy.z |

## 6.3.28 PI services

### Description

The PI_START function starts PI Services (Program Invocation Services) from the PLC in the NC area.

**Note**

A list of the available PI services can be found in the Basic Functions Function Manual.

### Programming

| Syntax: | **PI_START**("*Transfer string*") | |
|---|---|---|
| Description: | Executes the PI service | |
| Parameter: | "Transfer string" | Unlike the OEM documentation, the transfer string should be entered in inverted commas. |

### Example

```
PI_START("/NC,001,_N_LOGOUT")
```

**Note**

Channel-dependent PI Services always refer to the current channel.

PI services of the tool functions (TO area) always refer to the TO area that is assigned to the current channel.

## 6.3.29 Reading (RNP) and writing (WNP) system or user variables

### Description

The RNP (Read NC PLC) command can be used to read NC or PLC variables or machine data.

### Programming

| Syntax: | **RNP** ("*System or user variable", value*) | |
|---|---|---|
| Description: | Reads NC or PLC variable or machine data | |
| Parameters: | System or user variable | Name of NC or PLC variable |
| | Value | Value that is to be written to the system or user variable.<br>If the value is a String type, it must be written in double quotation marks. |

### Example

```
VAR2=RNP("$AA_IN[2]")         ; Read NC variable
```

### Description

The WNP (Write NC PLC) command can be used to write NC or PLC variables or machine data.

NC/PLC variables are accessed anew every time the WNP function is executed, i.e., NC/PLC access is always executed in a CHANGE method. It is advisable to use this option in cases where a system or user variable changes value frequently. If an NC/PLC variable is to be accessed only once, then it must be configured in a LOAD or UNLOAD method.

### Programming

| Syntax: | **WNP***("System or user variable", value)* | |
|---|---|---|
| Description: | Writes NC or PLC variable or machine data | |
| Parameters: | System or user variable | Name of NC or PLC variable |
| | Value | Value that is to be written to the system or user variable.<br>If the value is a String type, it must be written in double quotation marks. |

### Example

```
WNP("DB20.DBB1",1)        ; Write PLC variable
```

## 6.3.30 RESIZE_VAR_IO and RESIZE_VAR_TXT

### Description

The `RESIZE_VAR_IO()` and `RESIZE_VAR_TXT()` commands changes the geometry of the input/output component or text component of a variable.

After setting the new geometry, all fields of the screen are positioned as if the screen had been configured with these positions from the start. In this way, all fields are correctly aligned, depending on the configuration.

### Programming

| | | |
|---|---|---|
| Syntax: | **RESIZE_VAR_IO***(Variable name, [X], [Y], [Width], [Height])*<br>**RESIZE_VAR_TXT***(Variable name, [X], [Y], [Width], [Height])* | |
| Description: | Changes the geometry of the input/output component or text component of a variable. | |
| Parameter: | Variable name | Name of the variable whose geometry of the input/output component or text component of a variable is to be changed. |
| | X | X-coordinate, top left |
| | Y | Y-coordinate, top left |
| | Width | Width |
| | Height | Height |

**Note**

The previous value is retained for each value of X, Y, Width or Height that is not specified. If -1 is specified for one of these values, "Run MyScreens" sets the specified component of the default position.

### Example

```
RESIZE_VAR_IO("MyVar1", 200, , 100)    ; The I/O component of the "MyVar1" variable
                                       is moved to the X position 200 pixels, the
                                       width is set to 100 pixels. The Y position and
                                       the height of the previous value are retained.
```

## 6.3.31 Register (REG)

### Register description

Registers are needed in order to exchange data between different dialogs. Registers are assigned to each dialog. These are created when the first dialog is loaded and assigned the value 0 or an empty string.

**Note**

Registers must not be used directly in an OUTPUT method for generating NC code.

### Programming

| | | |
|---|---|---|
| Syntax: | **REG***[x]* | |
| Description: | Defines a register | |
| Parameters: | x | Register index with x = 0...19;<br>Type: REAL or STRING = VARIANT<br>Registers with x ≥ 20 have already been assigned by Siemens. |

### Description of register value

The assignment of values to registers is configured in a method.

**Note**

If a new dialog is generated from an existing dialog by means of the LM function, register content is automatically transferred to the new dialog at the same time and is available for further calculations in the second dialog.

### Programming

| | | |
|---|---|---|
| Syntax: | *Identifier*.val = *Register value*<br>or<br>*Identifier* = *Register value* | |
| Description: | | |
| Parameters: | Identifier | Name of the register |
| | Register value | Value of register |

### Example

```
UNLOAD
  REG[0] = VAR1          ; Assign value of variable 1 to register 0
```

```
END_UNLOAD

UNLOAD
  REG[9].VAL = 84        ; Assign value 84 to register 9
END_UNLOAD
                         ; These registers can then be assigned to local variables
                         again in a method in the next dialog.
LOAD
  VAR2 = REG[0]
END_LOAD
```

### Description of register status

The Status property can be used to scan a register for valid content.

One possible use for the register scan function is to ensure that a value is written to a register only if the relevant dialog is a "master dialog".

### Programming

| Syntax: | *Identifier*.vld | |
|---|---|---|
| Description: | The property is read-only property. | |
| Parameters: | Identifier | Name of the register |
| Return value: | | The result of the scan can be: |
| | FALSE = | invalid value |
| | TRUE = | valid value |

### Example

```
IF REG[15].VLD == FALSE    ; Scan validity of register value
  REG[15] = 84
ENDIF
VAR1 = REG[9].VLD          ; Assign the value of the REG[9] status request to Var1.
```

## 6.3.32 RETURN

### Description

The RETURN function can be used to prematurely terminate execution of the current subprogram and to return to the branch point of the last CALL command.

If no RETURN command is configured in the subprogram, the subprogram will run to the end before returning to the branch point.

### Programming

| Syntax: | **RETURN** | |
|---|---|---|
| Description: | Returns to the branch point | |
| Parameters: | - None - | |

### Example

```
//B(PROG1)                ; Block start
SUB(UP2)                  ; Start of subprogram
  IF VAR1.val=="Otto"
    VAR1.val="Hans"
    RETURN                ; If the variable value = Otto, the value "Hans" is as-
                          signed to the variable, and the subprogram ends at this
                          point.
  ENDIF
  VAR1.val="Otto"         ; If the variable value ≠ Otto, the value "Otto" is as-
                          signed to the variable.
END_SUB                   ; End of subprogram
//END                     ; Block end
```

## 6.3.33 Recompile

### Description

In the programming support system, it is possible to **recompile** NC code that has been generated with the GC function and to display the variable values in the input/output field of the associated entry dialog again.

### Programming

Variables from the NC code are transferred to the dialog. At the same time, the variable values from the NC code are compared with the calculated variable values from the configuration file. If the values do not coincide, an error message is written to the log book because values have been changed during NC code generation.

If the NC code contains the same variable several times, it is evaluated at the point where it last occurs during recompilation. A warning is also written to the log book.

Variables not utilized in NC code during code generation are stored as user comment. The term "user comment" refers to all information required to recompile codes. User comment must not be altered.

**Note**

The block consisting of NC code and user comment can be recompiled only if it starts at the beginning of a line.

**Examples:**

The programm contains the following NC code:

```
DEF VAR1=(I//101)
OUTPUT(CODE1)
  "X" VAR1 " Y200"
  "X" VAR1 " Y0"
END_OUTPUT
```

The following code is then stored in the parts program:

```
;NCG#TestGC#\cus.dir\aeditor.com#CODE1#1#3#
X101 Y200
X101 Y0
;#END#
```

The Editor reads the following during recompilation:

```
X101 Y200
X222 Y0          ; The value for X has been changed in the part program (X101 → X222)
```

The following value is displayed for VAR1 in the input dialog: `VAR1 = 222`

## See also

Generate code (GC) (Page 129)

## 6.3.34 Recompile without user comment

## Description

In the Programming Support, it is possible to **recompile without user comments** the NC code that has been generated with the GC function and to display the variable values in the input/ output field of the associated entry dialog again.

## Programming

The GC command can be executed in the following way in order to suppress comment lines that are generated for standard code generation:

```
GC("CODE1",D_NAME,1)
```

Normally, the resulting code cannot be recompiled. The following steps are required in order to be able to recompile the cycle calls generated in this way:

- **Expanding the easyscreen.ini**
  Section `[RECOMPILE_INFO_FILES]` will be introduced into the easyscreen.ini file. In this section, all ini files are listed that contain descriptions for cycles recompiled without user comment:
  ```
   [RECOMPILE_INFO_FILES]
  IniFile01 = cycles1.ini
  IniFile02 = cycles2.ini
  ```
  Several ini files can be specified, whose names can be freely selected.
- **Creating an ini file for a cycle description**
  Save the ini file with the cycle descriptions under the following path:
  *[System user directory]*/cfg
  *[System oem directory]*/cfg
  *[System add_on directory]*/cfg
  A separate section is required for each cycle. The section name corresponds to the name of the cycle:
  ```
  [Cycle123]
  Mname = TestGC
  Dname = testgc.com
  OUTPUT = Code1
  Anzp = 3
  Version = 0
  Code_type = 1
  Icon = cycle123.png
  Desc_Text = This is describing text
  ```

| | |
|---|---|
| Mname | Screen form name |
| Dname | Name of the file in which the screen is defined |
| OUTPUT | Name of the relevant OUTPUT method |
| Anzp | Number of parameters of the screen to be recompiled (all with DEF-created variables, also help variables) |
| Version | (optional) version specification for cycle |

| | |
|---|---|
| Icon | (optional) icon for display in the machining step program, format *.png<br>Screen size for corresponding resolution:<br>640 X 480 mm → 16 x 16 pixels<br>800 X 600 mm → 20 x 20 pixels<br>1024 X 768 mm → 26 x 26 pixels<br>1280 X 1024 mm → 26 x 26 pixels<br>1280 X 768 mm → 26 x 26 pixels<br>File loc.: *[System user directory]*ico/ico<Resolution><br>**Note:** For resolutions of 1280, the folder for 1024 x 768 mm is used (only suitable for machining step programs). |
| Desc_Text | (optional) Explanation text for display in the machining step program, max. length of string is 17 characters (only suitable for machining step programs). |

## Example

```
//M(TestGC/"Code generation:")
DEF VAR1 = (R//1)
DEF VAR2 = (R//2)
DEF D_NAME
LOAD
  VAR1 = 123
  VAR2 = -6
END_LOAD
OUTPUT(CODE1)
  "Cycle123(" VAR1 "," VAR2 ")"
  "M30"
END_OUTPUT

PRESS(VS1)
  D_NAME = "\MPF.DIR\MESSEN.MPF"
  GC("CODE1",D_NAME)                    ;Write NC code from the OUTPUT method to
                                        file \MPF.DIR\MESSEN.MPF:
                                        Cycle123(123, -6)
                                        M30
END_PRESS
```

### Supplementary conditions

- The "Recompile without user comment" functionality does not have the full range of functions as "Recompile with user comment".
  Typical cycle calls, such as MYCYCLE(PAR1, PAR2, PAR3, …), are supported during "Recompile without user comment". However, there must not be a user comment in the line of the function call. Optional parameters that are not transferred during the function and are of the string S type, must however be at least specified with empty quotation marks, e.g. "". Otherwise, "Run MyScreens" tries to fill in these parameters with commas and then recompiles the "filled-in cycle call".
- Parameters of the string type must not have any commas or semicolons in the string to be transferred.
- During "Recompile without user comment", all variables contained in the OUTPUT method must always be within the brackets so that the "Fill in missing cycle parameters with commas" functionality can take effect.

Example:

Permitted:

```
OUTPUT
    "MYCYCLE(" MYPAR1 "," MYPAR2 "," MYPAR3 ")"
END_OUTPUT
```

Not permitted (the MYCOMMENT variable is positioned after the closing bracket):

```
OUTPUT
    "MYCYCLE(" MYPAR1 "," MYPAR2 "," MYPAR3 ")" MYCOMMENT
END_OUTPUT
```

## 6.3.35 Search Forward, Search Backward (SF, SB)

### Description

The **SF, SB (Search Forward, Search Backward)** function is used to search for a string from the current cursor position in the NC program currently selected in the Editor and to output its value.

### Programming

| | |
|---|---|
| Syntax: | **SF***("String")* |
| Identifiers: | **S**earch **F**orward: Search forward from the current cursor position |
| Syntax: | **SB***("String")* |

| | | |
|---|---|---|
| Identifiers: | **S**earch **B**ackward: Search backward from the current cursor position | |
| Parameters: | String | Text to be found |

### Rules governing text search

- A blank must be inserted before and after the search concept unit, consisting of search string and its value, in the currently selected NC program.
- The system does not search for concepts within comment text or other strings.
- The value to be output must be a numerical expression. Expressions in the form of "X1=4+5" are not recognized.
- The system recognizes hexadecimal constants in the form of X1='HFFFF', binary constants in the form of X1='B10010' and exponential components in the form of X1='-.5EX-4'.
- The value of a string can be output if it contains the following between string and value:
  - Nothing
  - Blanks
  - Equality sign

### Example

The following notations are possible:

```
X100 Y200               ; The variable Abc is assigned the value 200
Abc = SB("Y")
X100 Y 200              ; The variable Abc is assigned the value 200
Abc = SB("Y")
X100 Y=200              ; The variable Abc is assigned the value 200
Abc = SB("Y")
```

## 6.3.36 STRING functions

### Overview

The following functions enable strings to be processed:

- Determine length of string
- Find a character in a string
- Extract substring from left
- Extract substring from right
- Extract substring from mid-string
- Replace substring

- Compare strings
- Insert a string in another string
- Remove a string from a string
- Remove blanks (from the left or from the right)
- Insert values or strings with formatting identifiers

## LEN function: Length of a string

| Syntax: | **LEN***(string \| varname)* | |
|---|---|---|
| Description: | Determines the number of characters in a string | |
| Parameters: | string | Every valid string expression. NULL is output if string is empty. |
| | varname | Any valid declared variable name |
| | Only one of the two parameters is allowed. | |

## Example

```
DEF VAR01
DEF VAR02

LOAD
  VAR01="HELLO"
  VAR02=LEN(VAR01)           ; Result = 5
END_LOAD
```

## INSTR function: Search for character in string

| Syntax: | **INSTR***(Start, String1, String2 [,Direction])* | |
|---|---|---|
| Description: | Searches for characters | |
| Parameters: | Start | Starting position for searching for string1 in string2. Enter 0 to start searching at the beginning of string2. |
| | String1 | Character that is being searched for. |
| | String2 | Character string in which the search is being made. |
| | Direction (optional) | Direction in which the search is being made.<br>0: From left to right (default setting)<br>1: From right to left |
| | 0 is returned if string1 does not occur in string2. | |

## Example

```
DEF VAR01
DEF VAR02

LOAD
  VAR01="HELLO/WORLD"
  VAR02=INST(1,"/",VAR01)        ; Result = 6
END_LOAD
```

## LEFT Function: String from left

| Syntax: | **LEFT***(string, length)* | |
|---|---|---|
| Description: | LEFT returns a string containing the specified number of characters starting from the left-hand side of a string. | |
| Parameters: | string | Character string or variable with the character string to be processed |
| | length | Number of characters that are to be read out |

## Example

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="HELLO/WORLD"
  VAR02=LEFT(VAR01,5)           ; Result = "HELLO"
END_LOAD
```

## RIGHT function: String from right

| Syntax: | **RIGHT***(string, length)* | |
|---|---|---|
| Description: | RIGHT returns a string containing the specified number of characters starting from the right-hand side of a string. | |
| Parameters: | string | Character string or variable with the character string to be processed |
| | length | Number of characters that are to be read out |

## Example

```
DEF VAR01
DEF VAR02
```

```
LOAD
  VAR01="HELLO/WORLD"
  VAR02=LEFT(VAR01,4)         ; Result = "WORLD"
END_LOAD
```

## MIDS function: String from mid-string

| Syntax: | **MIDS***(string, start [, length])* | |
|---|---|---|
| Description: | MIDS returns a string containing the specified number of characters starting at the specified position in the string. | |
| Parameters: | string | Character string or variable with the character string to be processed |
| | start | Start from where characters are to be read in the string |
| | length | Number of characters that are to be read out |

## Example

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="HELLO/WORLD"
  VAR02=LEFT(VAR01,4,4)            ; Result = "LO/W"
END_LOAD
```

## REPLACE Function: Replacing characters

| Syntax: | **REPLACE***( string, FindString, ReplaceString [, start [, count ] ] )* | |
|---|---|---|
| Description: | REPLACE replaces a character/string in a string with another character/string. | |
| Parameters: | string | String in which FindString is to be replaced with Replace-String. |
| | FindString | String to be replaced |
| | ReplaceString | Replacement string (is used instead of the FindString) |
| | start | Starting position for search and replace operations |
| | count | Number of characters that are to be searched from the start-ing position after FindString |
| Return value: | | |
| | string = Empty string | Copy of string |
| | FindString = Empty string | Copy of string |
| | ReplaceString = Empty string | Copy of string, in which all occurrences of Find-String are deleted |
| | start > Len(String) | Empty string |
| | count = 0 | Copy of string |

## STRCMP function: Compare strings

| Syntax: | **STRCMP***(string1, string2 [, CaseInsensitive])* | |
|---|---|---|
| Description: | STRCMP compares a character string with another character string. | |
| Parameters: | string1 | Character string that is to be compared with a second character string |
| | string2 | Character string that is to be compared with the first character string |
| | CaseInsensitive | Comparison with/without consideration of upper/lower case:<br>Not specified or FALSE = upper/lower case is considered<br>TRUE = upper/lower case is not considered |

## Example

```
REG[0]=STRCMP("Hugo", "HUGO")              ; Result = 32
                                           ; (<> 0, strings are not identical)

REG[0]=STRCMP("Hugo", "HUGO", TRUE)        ; Result = 0
                                           ; (== 0, stings are identical)
```

## STRINSERT function: Insert string

| Syntax: | **STRINSERT***(string1, string2 , insert position)* | |
|---|---|---|
| Description: | STRINSERT inserts a character string at a specific position in a character string. | |
| Parameters: | string1 | Character string in which a character string is to be inserted |
| | string2 | Character string that is to be inserted |
| | insert position | Position at which the character string is to be inserted |

## Example

```
REG[0]=STRINSERT("Hello!", " world", 5)    ; Result = "Hello world!")
```

## STRREMOVE function: Remove string

| Syntax: | **STRREMOVE***(string1, remove positon , count)* |
|---|---|
| Description: | STRREMOVE removes a specific number of characters at a specific position within a character string. |

| | | |
|---|---|---|
| Parameters: | string1 | Character string from which characters are to be removed |
| | remove position | Position at which characters are to be removed from the character string |
| | count | Number of characters to be removed |

### Example

```
REG[0]=STRREMOVE("Hello world!", 5, 6)     ; Result = "Hello!")
```

## TRIMLEFT function: Remove blanks from the left of the string

| | | |
|---|---|---|
| Syntax: | **TRIMLEFT***(string1)* | |
| Description: | TRIMLEFT removes blanks on the left from a character string. | |
| Parameters: | string1 | Character string from which blanks are to be removed from the left of the character string |

### Example

```
REG[0]=TRIMLEFT("    Hello!")              ; Result = "Hello!")
```

## TRIMRIGHT function: Remove blanks from the right of the string

| | | |
|---|---|---|
| Syntax: | **TRIMRIGHT***(string1)* | |
| Description: | TRIMLEFT removes blanks on the right from a character string. | |
| Parameters: | string1 | Character string from which blanks are to be removed from the right of the character string |

### Example

```
REG[0]=TRIMRIGHT("Hello!    ")             ; Result = "Hello!")
```

## FORMAT function: Insert values or string with formatting identifier

| | |
|---|---|
| Syntax: | **FORMAT***(Text with formatting identifier [, Value1] ... [, Value28] )* |
| Description: | Through the use of formatting identifiers, the FORMAT function enables up to 28 values or strings to be inserted at specified positions in a specific text. |

| | | |
|---|---|---|
| Formatting identifiers: | Syntax | %[Flags] [Width] [.decimal places] type |
| | Flags | Optional character for defining output formatting:<br>• Right-justified or left-justified ("-") for left-justified<br>• Add leading zeros ("0")<br>• default: Fill in with blanks if the value to be output has fewer positions than specified with [Width]. |
| | Width | The argument defines the minimum output width for a non-negative number. If the value has fewer positions than specified by the argument, the missing spaces are filled with blanks. |
| | Decimal places | With floating-point numbers, the optional parameter defines the number of decimal places. |
| | Type | The type character defines which data formats are transferred to the Print statement.<br>These characters need to be specified.<br>• d: Integer value<br>• f: Floating-point number<br>• s: String<br>• o: Octal<br>• x: Hexadecimal<br>• b: Binary |

## Example

```
DEF VAR1
DEF VAR2
LOAD
VAR1 = 123
VAR2 = FORMAT("Hello %08b %.2f %s!", VAR1 + 1, 987.654321, "world")
                                    ; Result = "Hello 01111100 987.65 world!"
END_LOAD
```

## See also

Use of strings (Page 80)

## 6.3.37 WHILE/UNTIL loops

## Description

A loop can be implemented with the DO LOOP commands. Depending on the configuration, it is run through as long as a condition is satisfied (WHILE) or until a condition occurs (UNTIL).

As loops can impair the system performance, depending on the configuration, they should be used carefully and without time-intensive actions in the loops.

It is recommended that a register (REG[]) be used as run variable, because normal display variables (particularly those with system or user variables connection) also impair the system performance due to the frequent updates or write operations.

The runtime of "Run MyScreens" methods can be determined with the DEBUG function (see Section DEBUG (Page 121)). Problems caused by loops (high CPU load, reduced response capability) may be identified in this way.

**Note**

As each FOR loop can be replaced by a WHILE loop, the syntax to formulate a FOR loop is not supported in EasyScreen.

## Programming

```
DO
   <instructions>
LOOP_WHILE <Condition to continue the loop>

DO
   <instructions>
LOOP_UNTIL <Condition to terminate the loop>

DO_WHILE <Condition to continue the loop>
   <instructions>
LOOP

DO_UNTIL <Condition to terminate the loop>
   <instructions>
LOOP
```

## Example

```
  REG[0] = 5
DO
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
     REG[1] = -5

   DO
      DEBUG("INNER: " << REG[1])
```

```
      REG[1] = REG[1] + 1
   LOOP_WHILE REG[1] < 0

LOOP_WHILE REG[0] < 10
```

```
  REG[0] = 5
DO
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
     REG[1] = -5

   DO
      DEBUG("INNER: " << REG[1])
      REG[1] = REG[1] + 1
   LOOP_UNTIL 0 <= REG[1]

LOOP_WHILE 10 > REG[0]
```

```
  REG[0] = 5
DO_WHILE 10 > REG[0]
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
     REG[1] = -5

   DO_UNTIL 0 <= REG[1]
     DEBUG("INNER: " << REG[1])
     REG[1] = REG[1] + 1
   LOOP

LOOP
```

```
  REG[0] = 5
DO_WHILE 10 > REG[0]
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
     REG[1] = -5

   DO
     DEBUG("INNER: " << REG[1])
```

```
     REG[1] = REG[1] + 1
   LOOP_UNTIL 0 <= REG[1]

LOOP
```

## 6.3.38 Cyclic execution of scripts: START_TIMER, STOP_TIMER

### Description

SUB methods can be called cyclically with the aid of timers. The START_TIMER() and STOP_TIMER() functions are available for this purpose.

**Note**

Only one timer can be configured for each SUB method.

### Programming

| Syntax: | **START_TIMER***("SUB name", Interval)* | |
|---|---|---|
| Description: | Starts cyclic processing of a SUB method | |
| Parameters: | SUB name: | Name of the SUB method to be called cyclically |
| | Interval: | Interval in milliseconds |

| Syntax: | **STOP_TIMER***("SUB name", Interval)* | |
|---|---|---|
| Description: | Stops cyclic processing of a SUB method | |
| Parameters: | SUB name: | Name of the SUB method whose timer is to be stopped. |

### Example

```
//M(TimerSample/"My timer")
DEF MyVariable=(I//0/,"Number of cyclic calls:"/WR1)
VS1=("Start%ntimer")
VS2=("Stop%ntimer")

SUB(MyTimerSub)
   MyVariable = MyVariable + 1
END_SUB
```

```
//M(TimerSample/"My timer")
PRESS(VS1)
   ;Calls SUB "MyTimerSub" every 1000 milliseconds
   START_TIMER("MyTimerSub", 1000)
END_PRESS

PRESS(VS2)
   STOP_TIMER("MyTimerSub")
END_PRESS
```

If START_TIMER is called again for a timer already assigned to a SUB method, then the new interval is taken over if it is different. Otherwise, the second call is ignored.

The smallest interval for the system is100 milliseconds.

If STOP_TIMER is called for a SUB method for which no timer is currently running, the call is ignored.

# Graphic and logic elements 7

## 7.1 Line, dividing line, rectangle, circle and ellipse

### Description

Lines, dividing lines, rectangles and ellipses/circles are configured in the LOAD method:

- Transparent rectangles are created by setting the fill color to the system background color.
- With the ELLIPSE element, a circle is created by setting the height and the width to the same value.
- Horizontal and vertical dividing lines always have the exact window width or window height. No scrollbars are produced by dividing lines.
  If you previously used the RECT element to display dividing lines, e.g. RECT(305,0,1,370,0,0,1), this is detected by the Programming Support and automatically converted to V_SEPARATOR(305,1,"#87a5cd", 1). This applies to both vertical and horizontal dividing lines.

### LINE element

Programming:

| | | |
|---|---|---|
| Syntax: | LINE (x1,y1,x2,y2,f,s) | |
| Description: | Defines a line | |
| Parameter: | x1 | Start point x-coordinate |
| | y1 | Start point y-coordinate |
| | x2 | End point x-coordinate |
| | y2 | End point y-coordinate |
| | f | Color of the line |
| | s | Line style:<br>1 = solid<br>2 = dashed<br>3 = dotted<br>4 = dashed and dotted |

### RECT element

Programming:

| | |
|---|---|
| Syntax: | RECT (x,y,w,h,f1,f2,s) |
| Description: | Defines a rectangle |

| | | |
|---|---|---|
| Parameter: | x | x-coordinate, top left |
| | y | y-coordinate, top left |
| | w | Width |
| | h | Height |
| | f1 | Color of the border |
| | f2 | Fill color |
| | s | Border style:<br>1 = solid<br>2 = dashed<br>3 = dotted<br>4 = dashed and dotted |

## ELLIPSE element

Programming:

| | | |
|---|---|---|
| Syntax: | ELLIPSE(x,y,w,h,f1,f2,s) | |
| Description: | Defines an ellipse or circle | |
| Parameter: | x | x-coordinate, top left |
| | y | y-coordinate, top left |
| | w | Width |
| | h | Height |
| | f1 | Color of the border |
| | f2 | Fill color |
| | s | Border style:<br>1 = solid<br>2 = dashed<br>3 = dotted<br>4 = dashed and dotted |

A circle is created when the values for height and width are the same.

## V_SEPARATOR element

Programming:

| | | |
|---|---|---|
| Syntax: | V_SEPARATOR (x,w,color,pen) | |
| Description: | Defines a vertical dividing line | |
| Parameter: | x | x position |
| | w | Line weight |
| | color | Color |
| | pen | Line style:<br>1 = solid<br>2 = dashed<br>3 = dotted<br>4 = dashed and dotted |

### H_SEPARATOR element

Programming:

| | | |
|---|---|---|
| Syntax: | H_SEPARATOR (y,h,color,pen) | |
| Description: | Defines a horizontal dividing line | |
| Parameter: | y | y position |
| | h | Line weight |
| | color | Color |
| | pen | Line style:<br>1 = solid<br>2 = dashed<br>3 = dotted<br>4 = dashed and dotted |

### See also

LOAD (Page 100)

### See also

CLEAR_BACKGROUND (Page 113)

## 7.2 Defining an array

### Definition

An array can be used to organize data of the same data type stored in the memory in such a way that it is possible to access the data via an index.

### Description

Arrays can be one- or two-dimensional. A one-dimensional array is treated like a two-dimensional array with just one line or column.

Arrays have start identifier //A and end identifier //END. The number of lines and columns is optional. An array is structured in the following way:

### Programming

| | | |
|---|---|---|
| Syntax: | ***//A****(Identifier)*<br>(a/b...)<br>(c/d...)<br>...<br>//END | |
| Description: | Defines array | |
| Parameters: | Identifier | Name of array |
| | a, b, c, d | Values of array<br>Values of the STRING type must be enclosed in double quotation marks. |

### Example

```
//A(Thread)                      ; Size/lead/core diameter
(0.3 / 0.075 / 0.202)
(0.4 / 0.1   / 0.270)
(0.5 / 0.125 / 0.338)
(0.6 / 0.15  / 0.406)
(0.8 / 0.2   / 0.540)
(1.0 / 0.25  / 0.676)
(1.2 / 0.25  / 0.676)
(1.4 / 0.3   / 1.010)
(1.7 / 0.35  / 1.246)
//END
```

## 7.2.1 Accessing the value of an array element

### Description

The value of an array access operation can be transferred with property Value (identifier.val).

The line index (line number of the array) and the column index (column number of the array) each begin at 0. If a line index or column index is outside the array, the value 0 or a blank string is output and the ERR variable is set to TRUE. Variable ERR is also set to TRUE if a search concept cannot be found.

### Programming

| | |
|---|---|
| Syntax: | Identifier [Z,[M[,C]]].val or<br>Identifier [Z,[M[,C]]] |
| Description: | Access to one-dimensional array with only one column |

| | | | |
|---|---|---|---|
| Syntax: | Identifier [S,[M[,C]]].val] or<br>Identifier [S,[M[,C]]] or | | |
| Description: | Access to one-dimensional array with only one line | | |
| Syntax: | Identifier [Z,S,[M[,C]]].val or<br>Identifier [Z,S,[M[,C]]] | | |
| Description: | Access to two-dimensional array | | |
| Parameters: | Identifier: | Name of array | |
| | Z: | Line value (line index or search concept) | |
| | S: | Column value (column index or search concept) | |
| | M: | Access mode | |
| | | 0 | Direct |
| | | 1 | Searches the line, column directly |
| | | 2 | Searches the column, line directly |
| | | 3 | Searches |
| | | 4 | Searches line index |
| | | 5 | Searches column index |
| | C: | Compare mode | |
| | | 0 | Search concept must be located in the range of values of the line or column. |
| | | 1 | Search concept must be located exactly. |
| Example: | `VAR1 = MET_G[REG[3],1,0].VAL` | `;Assign Var1 a value from array MET_G` | |

## Access mode

- **"Direct" access mode**
  With "Direct" access mode (M = 0), the array is accessed with the line index in Z and the column index in S. Compare mode C is not evaluated.
- **"Search" access mode**
  In the case of access mode M = 1, 2 or 3, the search always commences in line 0 or column 0.

| Mode M | Line value Z | Column value S | Output value |
|---|---|---|---|
| 0 | Line index | Column index | Value from line Z and column S |
| 1 | Search concept:<br>Search in column 0 | Column index of column from which value is read | Value from line found and column S |
| 2 | Line index of line from which return value is read | Search concept:<br>Search in line 0 | Value from line Z and column found |
| 3 | Search concept:<br>Search in column 0 | Search concept:<br>Search in line 0 | Value from line and column found |

| Mode M | Line value Z | Column value S | Output value |
|---|---|---|---|
| 4 | Search concept:<br>Search in column S | Column index of search column | Line index |
| 5 | Line index of search line. | Search concept:<br>Search in line Z | Column index |

### Compare mode

When compare mode C = 0 is used, the content of the search line or search column must be sorted in ascending order. If the search concept is smaller than the first element or greater than the last, the value 0 or a blank string is output and the error variable ERR is set to TRUE.

When compare mode C = 1 is used, the search concept must be present in the search line or search column. If the search concept cannot be found, the value 0 or a blank string is output and the error variable ERR is set to TRUE.

## 7.2.2 Example Access to an array element

### Prerequisite

Two arrays are defined below. These are the basis for the following examples:

```
//A(Thread)
                (0.3 / 0.075 / 0.202)
                (0.4 / 0.1   / 0.270)
                (0.5 / 0.125 / 0.338)
                (0.6 / 0.15  / 0.406)
                (0.8 / 0.2   / 0.540)
                (1.0 / 0.25  / 0.676)
                (1.2 / 0.25  / 0.676)
                (1.4 / 0.3   / 1.010)
                (1.7 / 0.35  / 1.246)
//END


//A(Array2)
                ("DES" /     "PTCH" /    "CDM" )
                (0.3 /       0.075 /     0.202 )
                (0.4 /       0.1 /       0.270 )
                (0.5 /       0.125 /     0.338 )
                (0.6 /       0.15 /      0.406 )
                (0.8 /       0.2 /       0.540 )
                (1.0 /       0.25 /      0.676 )
                (1.2 /       0.25 /      0.676 )
```

```
                (1.4 /       0.3 /      1.010 )
                (1.7 /       0.35 /     1.246 )
//END
```

## Examples

- **Access mode example 1:**
  The search concept is in Z. This key is always sought in column 0. The value from column S is output with the line index of the concept found.
  ```
  VAR1 = Thread[0.5,1,1]   ;VAR1 has the value 0.125
  ```
  Explanation:
  Search for value 0.5 in column 0 of "Thread" array and output the value found in column 1 of the same line.

- **Access mode example 2:**
  The search concept is in S. This concept is always searched for in line 0. The value from line Z is output with the column index of the concept found:
  ```
  VAR1 = ARRAY2[3,"PTCH",2]   ;VAR1 has the value 0.125
  ```
  Explanation:
  Search for column containing "PTCH" in line 0 of array "Array2". Output the value from the column found and the line with index 3.

- **Access mode example 3:**
  A search concept is in each of Z and S. The line index is searched for in column 0 with the concept in Z and the column index in line 0 with the concept in S. The value from the array is output with the line index and column index found:
  ```
  VAR1 = ARRAY2[0.6,"PTCH",3]   ;VAR1 has the value 0.15
  ```
  Explanation:
  Search for the line with the content 0.6 in column 0 of array "Array2", search for the column with the content "STG" in line 0 of Array2. Transfer the value from the line and column found to VAR1.

- **Access mode example 4:**
  The search concept is in Z. S contains the column index of the column in which concept is being searched for. The line index of the concept found is output:
  ```
  VAR1 = Thread[0.125,1,4]   ;VAR1 has the value 2
  ```
  Explanation:
  Search for value 0.125 in column 1 of array "Thread" and transfer the line index of the value found to VAR1.

- **Access mode example 5:**
  Z contains the line index of line in which concept is being searched for. The search concept is in S. The column index of the concept found is output:
  ```
  VAR1 = Thread[4,0.2,5,1]  ;VAR1 has the value 1
  ```
  Explanation:
  Search in line 4 of the "Thread" array for the value 0.2 and transfer the column index of the value found to VAR1. Comparison mode 1 was selected because the values of line 4 are not sorted in ascending order.

### 7.2.3 Scanning the status of an array element

#### Description

The Status property can be used to run a scan to find out whether an array access operation is supplying a valid value.

#### Programming

| | |
|---|---|
| Syntax: | *Identifier [Z, S, [M[,C]]]*.vld |
| Description: | Status is a read-only property. |
| Parameters: | Identifier Name of array |
| Return Value: | FALSE =invalid value |
| | TRUE = valid value |

#### Example

```
DEF MPIT = (R///"MPIT",,"MPIT",""/wr3)
DEF PIT  = (R///"PIT",,"PIT",""/wr3)
PRESS(VS1)
  MPIT = 0.6
  IF MET_G[MPIT,0,4,1].VLD == TRUE
    PIT    = MET_G[MPIT,1,0].VAL
    REG[4] = PIT
    REG[1] = "OK"
  ELSE
    REG[1] = "ERROR"
  ENDIF
END_PRESS
```

## 7.3 Table description (grid)

#### Definition

In contrast to the array, the values of a table (grid) are continually updated. It is a tabular display of values that can come from the NC or PLC. The table is organized in columns and always has variables of the same type in one column.

## Assignment

The reference to a table description is performed in the variables definition:

- The variables definition determines the values to be displayed, and the definition of table elements determines the appearance and arrangement on the screen window. The properties of the input/output fields from the definition line of the variables are taken over in the table.
- The visible area of the table is determined by the width and height of the input/output field. Any lines or columns than cannot be seen can be displayed by scrolling horizontally and vertically.

## Table identifiers

Identifiers of a table containing NCK or PLC values of the same type, which can be addressed via a channel block. The table identifier is differentiated from limits or toggle fields by the addition of a % sign in front of it. The file containing the table description can be specified by adding a comma after the identifier and then inserting the name of the file.

Identifier of a table for values of the same type, e.g. from the NCK or PLC. The table identifier is specified at the position at which the limit values or the toggle field are configured. The table identifier starts with a leading % character. This can be followed by a file name with a comma as separator. It specifies the file in which the table description is defined. Per default, the table is sought in the configuration file in which the screen is configured.

A table definition can also be loaded dynamically, e.g. in the LOAD method via the Load Grid (LG) function.

## System or user variable

This parameter remains empty for tables because the column definition lines contain detailed information about the variables to be displayed. The table description can be provided dynamically.

## Example

Definition of a MyGridVar variable which displays a "MyGrid1" grid in its input/output field (distance from left: 100, distance from top: Standard, width: 350, height: 100).

```
DEF MyGridVar=(V/% MyGrid1///////100,,350,100)
```

## See also

Parameters of variables (Page 69)

Load Grid (LG) (Page 137)

## 7.3.1 Defining a table (grid)

### Description

The table block comprises:

- Header
- 1 to n column descriptions

### Programming

| | | | |
|---|---|---|---|
| Syntax: | ***//G****(Table identifier/Table type/Number of lines/ [Fixed line attribute],[Fixed column attribute])* | | |
| Description: | Defines a table (grid) | | |
| Parameters: | Table identifier | The table identifier is used without a leading % sign. It can only be used once in a dialog. | |
| | Table type | 0 (default) | Table for PLC or user data (NCK- and channel-specific data) |
| | | 1 | and others, reserved |
| | Number of lines | Number of lines including header | |
| | | The fixed line or fixed column is not scrolled. The number of columns is the number of columns configured. | |
| | Display of column header line | -1: | The column header line is not displayed |
| | | 0: | The column header line is displayed (default) |
| | Number of fixed columns | 0: | No fixed column |
| | | 1 ... n: | Number of columns that are to be permanently visible, i.e. not scrolled horizontally |

### Examples

| | |
|---|---|
| `//G(grid1/0/5/-1,2)` | Column header line is hidden and two fixed columns |
| `//G(grid1/0/5/,1)` | Column header line is displayed and one fixed column |
| `//G(grid1/0/5)` | Column header line is displayed and no fixed columns |

## 7.3.2 Defining columns

### Description

For tables (grids), it is advisable to use variables with an index. For PLC or NC variables, the index number with one or more indices is of significance.

The values displayed in a table can be modified directly within the restrictions of the rights granted by the attributes and within any defined limits.

## Programming

| | | |
|---|---|---|
| Syntax: | *(Type/Limits/Empty/Long text,column header/Attributes/Help display/ System or user variable/Column width/Offset1, Offset2, Offset3)*<br>See also Extended configuration syntax (Page 36). | |
| Description: | Defines columns | |
| Parameters: | Similar to variables | |
| | Type | Data type |
| | Limits | Limit value MIN, limit value MAX |
| | Long text, column header | |
| | Attributes | |
| | Help display | |
| | System or user variable | As variable, PLC or NC variables should be entered in double quotation marks. |
| | Column width | Entry in pixels. |
| | Offset | The increment sizes to increment each index in order to fill the column are specified in the assigned offset parameter:<br>• Offset1: Step width for the 1st index<br>• Offset2: Step width for the 2nd index<br>• Offset3: Step width for the 3rd index |

## Column header from text file

The column header can be entered as text or text number ($8xxxx) and is not scrolled.

## Modifying column properties

The column properties which can be modified dynamically (written) are:

- Limits (min,max),
- Column header (st),
- Attributes (wr, ac and li),
- Help display (hlp) and
- OPI-Variable (var).

Column properties are modified via the variable identifier in the definition line and the column index (starting at 1).

Example: `VAR1[1].st="Column 1"`

Column properties cannot be read in the LOAD method.

The wr, ac and li attributes can be specified for column definitions.

### See also

Load Grid (LG) (Page 137)

## 7.3.3 Focus control in the table (cell)

### Description

The Row and Col properties can be used to set and calculate the focus within a table:

- Identifier.**Row**
- Identifier.**Col**

### Programming

Each cell in a table has the Val and Vld properties.

In order to read and write cell properties, a line and column index must be specified in addition to the variable identifiers from the definition list.

| | |
|---|---|
| Syntax: | Identifier[Line index, column index].val or<br>Identifier[Line index, column index] |
| Description: | Val properties |
| Syntax: | Identifier[Line index, column index].vld |
| Description: | Vld properties |

### Example

```
Var1[2,3].val=1.203
```

If the line and column indices are not specified, the indices of the focused cell apply. This corresponds to:

```
Var1.Row =2
Var1.Col=3
Var1.val=1.203
```

# 7.4 Custom widgets

## 7.4.1 Defining custom widgets

### Description

User-specific display elements are configured in the dialog using a custom widget.



**Software option**

In order to use custom widgets in dialog boxes, you require the following additional software option:

"SINUMERIK Integrate Run MyHMI /3GL" (6FC5800-0AP60-0YB0)

### Programming

| | | |
|---|---|---|
| Definition: | **DEF** *(name)* | |
| Syntax: | **(W///"","***(library name)***.***(class name)***"/////***a***,***b***,***c***,***d***);** | |
| Description: | W | Defining custom widgets |
| Parameters: | Name | Custom widget name, freely selectable |
| | Library name | Can be freely selected, name of the dll (Windows) or (Linux) library file |
| | Class name | Freely selectable, name of the class function from the previously named library |
| | a, b, c, d | Position and size of the configuration |

### Example

A custom widget is defined in the dialog configuration in the following way:

```
DEF Cus = (W///"","slestestcustomwidget.SlEsTestCustomWidget"/////
20,20,250,100);
```

## 7.4.2 Structure of the custom widget library

### Description

Essentially, the custom widget library contains a defined class. The name of this class must be specified in the dialog configuration in addition to the library names. Starting from library names, "Run MyScreens" accesses a dll file with the same name, e.g.:

```
slestestcustomwidget.dll
```

### Programming

The class definition of the dll file should look like this:

```
#define SLESTESTCUSTOMWIDGET_EXPORT Q_DECL_EXPORT


class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
 Q_OBJECT
....
....
}
```

## 7.4.3 Structure of the custom widget interface

### Description

The library is supplemented by an interface in order to display the custom widget in the dialog. This contains macro definitions with which "Run MyScreens" initiates the custom widget. The interface is available in the form of a cpp file. The file name can be freely selected, e.g.: `sleswidgetfactory.cpp`

### Programming

The interface is defined as follows:

```
#include "slestestcustomwidget.h"      ; The header file for the relevant custom widg-
                                       ets is inserted at the beginning of the file
....
//Makros                               ; Macro definitions are not changed
....
WIDGET_CLASS_EXPORT(SlEsTestCustom-    ; The relevant custom widget is declared at
Widget)                                the end of the file
```

### Example

Content of the file sleswidgetfactory.cpp for a custom widget with the class name SlEsTestCustomWidget":

```
#include <Qt/qglobal.h>
#include "slestestcustomwidget.h"


////////////////////////////////////////////////////////////////////////////
```

```
// MAKROS FOR PLUGIN DLL-EXPORT - DO NOT CHANGE
////////////////////////////////////////////////////////////////////////

#ifndef Q_EXTERN_C
#ifdef __cplusplus
#define Q_EXTERN_C    extern "C"
#else
#define Q_EXTERN_C    extern
#endif
#endif

#define SL_ES_FCT_NAME(PLUGIN) sl_es_create_ ##PLUGIN
#define SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( IMPLEMENTATION , PARAM) \
    { \
    IMPLEMENTATION *i = new PARAM; \
    return i; \
    }

#ifdef Q_WS_WIN
#   ifdef Q_CC_BOR
#       define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
    Q_EXTERN_C __declspec(dllexport) void* \
    __stdcall SL_ES_FCT_NAME(PLUGIN) (QWidget* pParent) \
    SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
#   else
#       define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
    Q_EXTERN_C __declspec(dllexport) void* SL_ES_FCT_NAME(PLUGIN) \
    (QWidget* pParent) \
    SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
#   endif
#else
#   define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
    Q_EXTERN_C void* SL_ES_FCT_NAME(PLUGIN) (QWidget* pParent) \
    SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
#endif

#define WIDGET_CLASS_EXPORT(CLASSNAME)  \
    EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(CLASSNAME,CLASSNAME(pParent))

////////////////////////////////////////////////////////////////////////
// FOR OEM USER - please declare here your widget classes for export
////////////////////////////////////////////////////////////////////////

WIDGET_CLASS_EXPORT(SlEsTestCustomWidget)
```

## 7.4.4 Interaction between custom widget and dialog box - Automatic data exchange

Custom widgets interact with dialog boxes and can display values or manipulate them.

### Conditions

Automatic data exchange takes place under the following conditions:

| Condition | Direction |
|---|---|
| When starting or recompiling a dialog | Dialog → custom widget |
| When executing the GC command for generating cycle calls | Custom widget → Dialog |

### Programming

The following definitions are necessary for the interaction:

**Expansion of the dialog configuration**

| Definition: | **DEF** *(variable)* | |
|---|---|---|
| Syntax: | **(***(type)***//5/"","***(variable)***","/wr2/)** | |
| Variable type: | Type | Standard input field (no grid or toggle) with any data type (no W) |
| Parameters: | Variable | Any designation of a variable for data exchange |
| Input mode: | wr2 | Reading and writing |

**Example**

```
DEF CUSVAR1 = (R//5/"","CUSVAR1",""/wr2/)
```

**Expansion of the class definition**

In the class definition of the custom widgets, a QProperty must be created whose name is identical to the selected variable of the dialog configuration, e.g.:

```
Q_PROPERTY(double CUSVAR1 READ cusVar1 WRITE setCusVar1);
```

**Example**

The class definition of the dll file should look like this:

```
#define SLESTESTCUSTOMWIDGET_EXPORT Q_DECL_EXPORT


class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
 Q_OBJECT
 Q_PROPERTY(double CUSVAR1  READ cusVar1    WRITE setCusVar1);
....
....
}
```

## 7.4.5 Interaction between custom widget and dialog box - Manual data exchange

Besides automatic data exchange, manual data exchange is also possible. Data exchange is dynamic, i.e. takes place while the dialog box is running. The following actions are possible:

- Properties of the custom widget can be read and written.
- Methods of the custom widget can be called from the Run MyScreens configuration.
- A response to a particular signal of the custom widget can be implemented in order to call subprograms (SUB) in the Run MyScreens configuration.

### 7.4.5.1 Reading and writing properties

#### Description

The functions ReadCWProperties and WriteCWProperties are provided in the Run MyScreens configuration for reading and writing properties of the custom widgets.

#### Programming

| | | |
|---|---|---|
| Syntax: | **ReadCWProperty***("Variablename", "Propertyname")* | |
| Description: | Reading a property of a custom widget | |
| Parameters: | Variablename | Name of a dialog box variable to which a custom widget is assigned |
| | Propertyname | Name of the custom widget property to be read |
| Return value: | Current value of the custom widget property | |

| | | |
|---|---|---|
| Syntax: | **WriteCWProperty***("Variablename", "Propertyname", "Value")* | |
| Description: | Writing the property of a custom widget | |
| Parameters: | Variable name | Name of a dialog box variable to which a custom widget is assigned |
| | Propertyname | Name of the custom widget property to be written |
| | Value | Value to be written to the property of the custom widget |

#### Examples

**Example 1:**

Read property "MyStringVar" of the custom widget that is linked to dialog box variable "MyCWVar1" and assign the value to register 7.

**Custom widget class declaration:**

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
   Q_PROPERTY(QString MyStringVar
              READ myStringVar
              WRITE setMyStringVar);
   ...
```

**Dialog box configuration:**

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
PRESS(VS1)
   REG[7]=ReadCWProperty("MyCWVar1", "MyStringVar")
END_PRESS
```

**Example 2:**

Write the result of the calculation "3 + sin(123.456)" into property "MyRealVar" of the custom widget that is linked to dialog box variable "MyCWVar1."

**Custom widget class declaration:**

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
   Q_PROPERTY(double MyRealVar
              READ myRealVar
              WRITE setMyRealVar);
   ...
```

**Dialog box configuration:**

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
PRESS(VS1)
   WriteCWProperty("MyCWVar1", "MyRealVar", 3 + sin(123.456))
END_PRESS
```

### 7.4.5.2 Executing a method of the custom widget

## Description

The function CallCWMethod is available in the Run MyScreens configuration for executing methods of the custom widget.

The custom widget method to be called must have no more than 10 transfer parameters.

The following transfer parameter data formats are supported:

- bool
- uint
- int
- double
- QString
- QByteArray

## Programming

| | | |
|---|---|---|
| Syntax: | **CallCWMethod***("Variablename", "Methodname[, Argument 0][, Argument 1 … [,Argument 9]")* | |
| Description: | Calling a custom widget method | |
| Parameters: | Variable name | Name of a dialog box variable to which a custom widget is assigned |
| | Method name | Name of the custom widget name to be called |
| | Argument 0 - 9 | Transfer parameter for the custom widget method<br>Supported data formats: see above<br>**Note:** The transfer parameters are always passed "By-Val," i.e. only the value and not, for example, the reference is passed to a variable. |
| Return value: | Return value of the custom widget method<br>The following transfer parameter data formats are supported:<br>• void<br>• bool<br>• uint<br>• int<br>• double<br>• QString<br>• QByteArray<br>**Notice:** Even if the data format of the return value of the custom widget method is "void," it must be formally assigned to a variable, for example. | |

## Example

### Custom widget class declaration:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
   public slots:
       void myFunc1(int nValue, const QString& szString, double dValue);
...
```

**Dialog box configuration:**

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
DEF MyStringVar1 = (S)
DEF MyRealVar = (R)


PRESS(VS3)
   REG[9] = CallCWMethod("MyCWVar1", "myFunc1", 1+7, MyStringVar1, sin(MyRealVar) -
8)
END_PRESS
```

### 7.4.5.3 Response to a custom widget signal

## Description

In Run MyScreens it is possible to respond to a particular signal (invokeSub()) of the custom widget and call up a subprogram (SUB).

10 global variables, the so-called SIGARG, are available for passing values (custom widget signal -> SUB), which are comparable to registers (REG) in configuration. This is where the values transferred with the custom widget signal are stored.

The following transfer parameter data formats are supported:

- bool
- uint
- int
- double
- QString
- QByteArray

## Programming

**Calling the subroutine:**

| | | |
|---|---|---|
| Syntax: | `void invokeSub(const QString& rszSignalName, const QVariantList& rvntList);` | |
| Description: | Custom widget signal with which a Run MyScreens subprogram is called. | |
| Parameters: | rszSignalName | Name of the Run MyScreens subprogram to be called |
| | rvntList | QVariantList array for transferring parameters that are stored in the global parameter SIGARG and which are available in the configuration.<br>Maximum size: 10 elements<br>Data formats supported: see above<br>**Note:** The transfer parameters are always passed "By-Val," i.e. only the value and not, for example, the reference to a variable is passed. |

### Subroutine to be called:

| | | |
|---|---|---|
| Syntax: | `SUB(on_<Variablename>_<Signalname>)`<br>`…`<br>`END_SUB` | |
| Description: | Response to a custom widget signal | |
| Parameters: | Variable name | Name of a dialog box variable to which a custom widget is assigned. |
| | Signal name | Name of the custom widget signal |
| | SIGARG 0 - 9 | Transfer parameters for the custom widget method.<br>Supported data formats: See above<br>**Notice:** The transfer parameters are always passed "By-Val," i.e. only the value and not, for example, the reference to a variable is passed. |

## Example

### Customer widget class declaration:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT signals: void invokeSub(const QString& szSubName, const QVariantList&
vntList);
…
}
```

### Custom widget class:

```
QVariantList vntList;
vntList << 123.456;
emit invokeSub("MySub", vntList);
```

### Dialog box configuration:

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
SUB(on_MyCWVar1_MySub)
   DEBUG("SUB(on_MyCWVar1_MySub) was called with parameter: """ << SIGARG[0] <<
""""")
END_SUB
```

### Result easyscreen_log.txt:

```
[10:22:40.445] DEBUG: SUB(on_MyCWVar1_MySub) was called with parameter: "123.456"
```

# "Custom" operating area

# 8

## 8.1 How to activate the "Custom" operating area

### Activating the "Custom" operating area

The "Custom" operating area is not activated on delivery.

1. First copy the slamconfig.ini. file from the *[System Siemens directory]*\cfg directory to the *[System oem directory]*\cfg directory
   or correspondingly to the *[System add_on directory]*\cfg directory or *[System user directory]*\cfg directory.
2. To activate the "Custom" operating area, the following must be entered:
   ```
   [Custom]
   Visible=True
   ```

### Result

After activation is complete, the softkey for the "Custom" operating area can be found in the main menu (F10) on the menu continuation bar on the HSK4 (= default).

The "Custom" operating area displays an empty window covering the entire operating area, with a configurable header. All horizontal and vertical softkeys can be configured.

## 8.2 How to configure the "Custom" softkey

### Configuring the softkey for the "Custom" operating area

The labeling and position of the softkey for the "Custom" operating area are configured in the slamconfig.ini file.

The following options are available for configuring the start softkey:

1. To replace a softkey label with a **language-dependent text**, the following must be entered in the [Custom] section:
   ```
   TextId=MY_TEXT_ID
   TextFile=mytextfile
   TextContext=mycontext
   ```
   In this example, the softkey shows the language-dependent text which was saved with the text ID "MY_TEXT_ID" in text file mytextfile_xxx.qm under "MyContext" (xxx stands for language code).
2. To replace a softkey label with a **language-neutral text**, the following must be entered in the [Custom] section:
   ```
   TextId=HELLO
   TextFile=<empty>
   TextContext=<empty>
   ```
   In this example, the softkey for the "Custom" operating area displays the text "HELLO" for every language.
3. **An icon** can also be displayed on the softkey in addition to the text.
   To do this, the following must be entered in the [Custom] section:
   ```
   Picture=mypicture.png
   ```
   The softkey then displays the icon from the file mypicture.png. Graphics and bitmaps are stored at the following path: *[System oem directory]*/ico/ico<Resolution>. The directory that corresponds to the display resolution must be used.
4. **The position** of the softkey can also be set. The following entry in the [Custom] section makes this setting:
   ```
   SoftkeyPosition=12
   ```
   The default is position 12. This corresponds to the HSK4 on the menu continuation bar of the operating area's menu. Positions 1 - 8 correspond to HSK1 to HSK8 on the menu bar, positions 9 - 16 to HSK1 to HSK8 on the menu continuation bar.

## 8.3 How to configure the "Custom" operating area

### Configuring the softkey for the "Custom" operating area

You need the easyscreen.ini and custom.ini files to configure the operating area. Templates for both these files are available in the *[System Siemens directory]*/templates/cfg directory.

1. First copy the files to the *[System oem directory]*/cfg directory and make your changes from there.
2. File easyscreen.ini already contains a definition line for the "Custom" operating area:
   ```
   ;StartFile02 = area := Custom, dialog := SlEsCustomDialog,
   startfile := custom.com
   ```
   The ";" at the start of the line represents the comment character. This means the line is commented out and, as such, not active. To change this, the ";" must be deleted.
   The "startfile" attribute in this line defines that the entry will refer to the custom.com project file when the "Custom" operating area is selected.

3. You create the **custom.com project file** in the *[System oem directory]*/proj directory. This contains the relevant configuration which is created in the same way as the aeditor.com file of the "Program" operating area. The configured start softkeys are then displayed in the "Custom" operating area.
4. You configure the **language-neutral text** for the title bar of the dialog in the custom.ini file. The following entry is available in the template for this purpose:
   ```
   [Header]
   Text=Custom
   ```
   You can replace this text with a customized one.
5. The template contains the following entry for configuring a **start screen** for the "Custom" operating area:
   ```
   [Picture]
   Picture=logo.png
   ```
   Logo.png is the name of the start screen which appears on the "Custom" operating area's start dialog. Here you can display a company logo, for example, or another image. The file should be saved in the directory for the corresponding resolution under: *[System oem directory]*/ico/ …
6. In order to display a specific "Run MyScreens" screen when the "Custom" operating area is shown the first time, the following entry is available in the template:
   ```
   ; Mask shown with startup of area "custom"
   [Startup]
   ;Startup = Mask:=MyCustomStartupMask, File:=mycustommasks.com
   ```
   Typically, SINUMERIK Operate is started directly with a specific "Run MyScreens" screen. The "startuparea" key must be set as follows in the "[miscellaneous]" section of the systemconfiguration.ini file:
   ```
   [miscellaneous]
   ;name of the area to be shown at system startup
   startuparea = Custom
   ```

## 8.4 Programming example for the "Custom" area

### Example

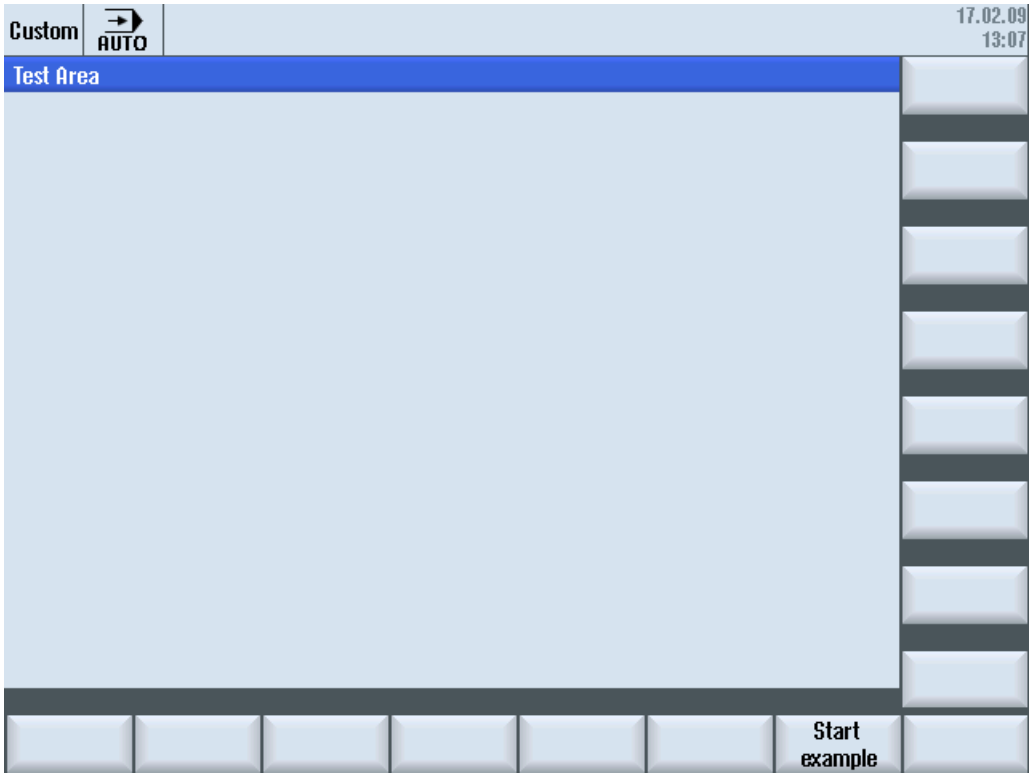



Figure 8-1 Example with "Start example" softkey

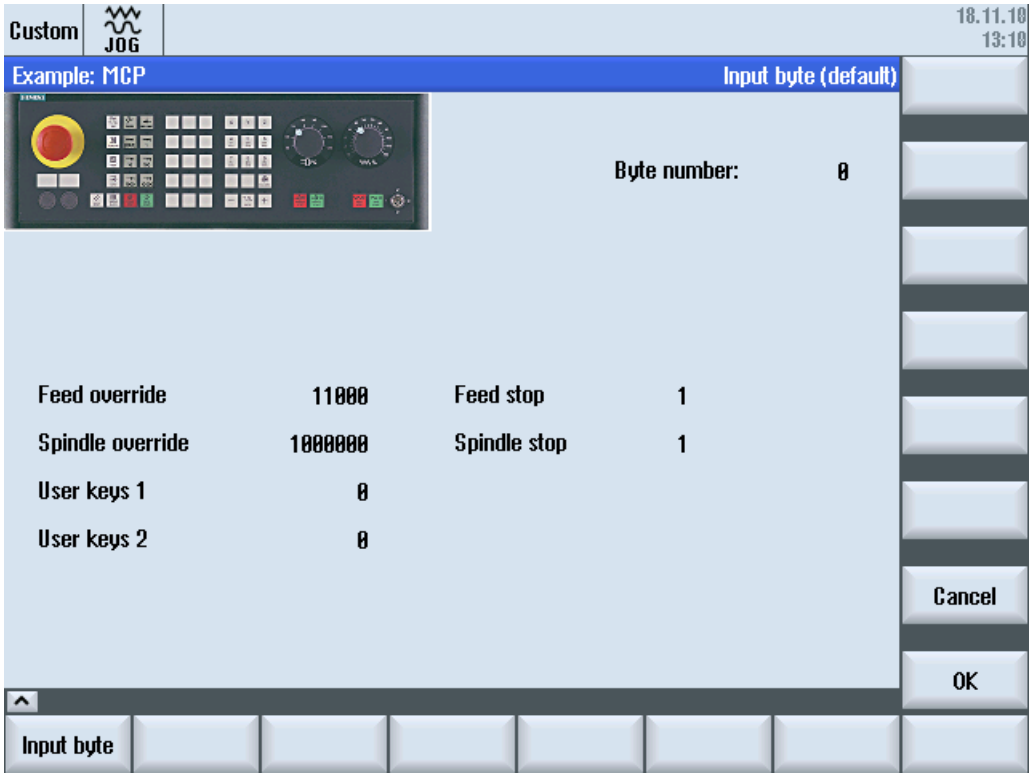



Figure 8-2 Example with bitmap and text fields

## File overview

The following files are required:

- custom.com
- easyscreen.ini

## Programming

Content of the custom.com file:

**Note**

The graphic file mcp.png integrated in the example is also only a sample file. If you wish to use this programming example in your application, you must replace the graphic by one of your own graphics.

```
//S(Start)
   HS7=("Start example", se1, ac7)
   PRESS(HS7)
      LM("Maske4")
```

```
   END_PRESS
//END
//M(Maske4/"Example: MCP"/"mcp.png")
   DEF byte=(I/0/0/"Input byte=0 (default)","Byte number:",""/wr1,li1///380,40,100/480,40,50)
   DEF Feed=(IBB//0/"","Feed override",""/wr1//"EB3"/20,180,100/130,180,100), Axistop=(B//0/"","Feed
stop",""/wr1//"E2.2"/280,180,100/380,180,50/100)
   DEF Spin=(IBB//0/"","Spindle override",""/wr1//"EB0"/20,210,100/130,210,100), spinstop=(B//
0/"","Spindle stop",""/wr1//"E2.4"/280,210,100/380,210,50/100)
   DEF custom1=(IBB//0/"," User keys 1",""/wr1//"EB7.7"/20,240,100/130,240,100)
   DEF custom2=(IBB//0/"","User keys 2",""/wr1//"EB7.5"/20,270,100/130,270,100)
   DEF By1
   DEF By2
   DEF By3
   DEF By6
   DEF By7

   HS1=("Input byte", SE1, AC4)
   HS2=("")
   HS3=("")
   HS4=("")
   HS5=("")
   HS6=("")
   HS7=("")
   HS8=("")
   VS1=("")
   VS2=("")
   VS3=("")
   VS4=("")
   VS5=("")
   VS6=("")
   VS7=("Cancel", SE1, AC7)
   VS8=("OK", SE1, AC7)

   PRESS(VS7)
      EXIT
   END_PRESS

   PRESS(VS8)
      EXIT
   END_PRESS

   LOAD
      By1=1
      By2=2
      By3=3
```

```
      By6=6
      By7=7
   END_LOAD

   PRESS(HS1)
      Byte.wr=2
   END_PRESS

   CHANGE(Byte)
      By1=byte+1
      By2=byte+2
      By3=byte+3
      By6=byte+6
      By7=byte+7
      Feed.VAR="EB"<<By3
      Spin.VAR="EB"<<Byte
      Custom1.VAR="EB"<<By6
      Custom2.VAR="EB"<<By7
      Axisstop.VAR="E"<<By2<<".2"
      Spinstop.VAR="E"<<By2<<".4"
      Byte.wr=1
   END_CHANGE

   CHANGE(Axis stop)
      IF Axistop==0
         Axistop.BC=9
      ELSE
         Axistop.BC=11
      ENDIF
   END_CHANGE

   CHANGE(Spin stop)
      IF Spinstop==0
         Spinstop.BC=9
      ELSE
         Spinstop.BC=11
      ENDIF
   END_CHANGE
//END
```

# Dialog selection 9

## 9.1 Dialog selection using PLC softkeys

### Configuration

Description of the procedure:

- The systemconfiguration.ini contains a section [keyconfiguration]. The entry specifies an action for a special PLC softkey.
- A number is given as an action. A "Run MyScreens" call is involved if the number is greater than or equal to 100.
- A section for defining the action to be performed must be created in the easyscreen.ini file. The name of the section is based on the name of the operating area and the dialog name (see entry under [keyconfiguration] → Area:=..., Dialog:=...) → [<Area>_<Dialog>]
  → e.g. [AreaParameter_SlPaDialog].
- The action numbers (which were given in the systemconfiguration.ini → see Action:=...) are defined in this section. There are two commands involved:
  1. LS("Softkey menu1","param.com") ... Loading a softkey menu
  2. LM("Screen form1","param.com") ... Loading a screen form

### Selecting softkey menus via PLC softkeys

"Run MyScreens" makes it possible to select "Run MyScreens" softkey menus and "Run MyScreens" dialogs via PLC softkeys. This can only be done if the "action" attribute to be specified when configuring the relevant PLC softkeys has a value greater than or equal to 100.

PLC softkeys are configured in the file `systemconfiguration.ini` in the section [keyconfiguration]:

```
[keyconfiguration]
KEY75.1 = Area:=area, Dialog:=dialog, Screen:=screen, Action:= 100,
```

The LM and LS commands to be executed upon activation of the relevant PLC softkeys are configured in the easyscreen.ini file. The names of the sections that are used for the purpose of configuration are structured as follows:

| | |
|---|---|
| `[areaname_dialogname]` | The first part of the name "areaname" refers to the operating area and the second part "dialogname" designates the dialog to which the commands configured in this section apply. |
| `[AreaParameter_SlPaDialog]`<br>`100.screen1 = LS("Softkey1","param.com")`<br>`101.screen3 = LM("Screen form1","param.com")` | The names given in the systemconfiguration.ini file for the operating area and dialog should be used. The dialog does not have to be specified.<br>This is particularly true for operating areas which are only implemented by means of a single dialog. Please refer to the example on the left.<br>If "screen1" is displayed in the AreaParameter operating area implemented by the SlPaDialog dialog, the "LS("Softkey1","param.com")" command will be executed when the "action" with the value 100 occurs. |
| `action.screen=Command` | Both the "action" and "screen" attributes clearly indicate when the specified command will be executed.<br>The "screen" information is optional.<br>The following commands are permissible:<br>`LM (LoadMask)`<br>`LS (LoadSoftkeys)` |

## 9.2 Dialog selection using PLC hard keys

### Field of application

The following functions can be initiated in the operating software by the PLC:

- Select an operating area
- Select certain scenarios within operating areas
- Execute functions configured at softkeys

### Hard keys

All keys - also the PLC keys - are subsequently referred to as hard keys. A maximum of 254 hard keys can be defined. The following allocation applies:

| Key number | Application |
|---|---|
| Key 1 – key 9 | Keys on the operator panel front |
| Key 10 – key 49 | Reserved |
| Key 50 – key 254<br>Key 50 – key 81 | PLC keys:<br>reserved for OEMs |
| Key 255 | Pre-assigned by control information. |

Hard keys 1 - 9 are pre-assigned as follows:

| Key designation | | Action / effect |
|---|---|---|
| HK1 | Position | Selects "Machine" operating area, last dialog |
| HK2 | Program | Selects "Program" operating area, last dialog or last program |
| HK3 | Offset | Selects "Parameter" operating area, last dialog |
| HK4 | Program manager | No function |
| HK5 | Alarm | Selects diagnostics operating area, "Alarm list" dialog |

## Configuration

The configuring is realized in the systemconfiguration.ini configuration file in the section [keyconfiguration]. Each line defines what is known as a hard key event. A hard key event is the n-th actuation of a specific hard key. For example, the second and third actuation of a specific hard key can result in different responses.

The entries in the systemconfiguration.ini configuration file can be overwritten with user-specific settings. The directories *[System user directory]*cfg and *[System oem directory]*cfg are available for this purpose.

The lines for configuring the hard key events have the following structure:

```
KEYx.n = Area:=area, Dialog:=dialog, Screen:=screen, Forms:=form,
Menus:=menu,
Action:=menu.action, Cmdline:=cmdline
KEYx.n = Area:=area, Dialog:=dialog, Cmdline:=cmdline, Action:=
action
```

x: Number of the hard key, range of values: 1 – 254

n: Event number – corresponds to the nth actuation of the hard key, value range: 0 – 9

## Requirement

The PLC user program must fulfill the following requirement:

Only one hard key is processed. As a consequence, a new request can only be set if the operating software has acknowledged the previous request. If the PLC user program derives the hard key from an MCP key, it must provide sufficient buffer storage of the key(s) to ensure that no fast keystrokes are lost.

## PLC interface

An area to select a hard key is provided in the PLC interface. The area is in DB19.DBB10. Here, the PLC can directly specify a key value of between 50 and 254.

Acknowledgment by the operating software takes place in two steps. This procedure is necessary so that the operating software can correctly identify two separate events if the same key code is entered twice consecutively. In the first step, control information 255 is written to byte DB19.DBB10. This defined virtual key activation enables the HMI to identify every PLC key sequence uniquely. The control information is of no significance to the PLC user program and must not be changed. In the second step, the actual acknowledgment takes place with

respect to the PLC by clearing DB19.DBB10. From this point in time, the PLC user program can specify a new hard key. In parallel, the actual hard key request is processed in the operating software.

## Example

Configuration file:

```
; PLC hard keys (KEY50-KEY254)
[keyconfiguration]
KEY50.0 = name := AreaMachine, dialog := SlMachine
KEY51.0 = name := AreaParameter, dialog := SlParameter
KEY52.0 = name := AreaProgramEdit, dialog := SlProgramEdit
KEY53.0 = name := AreaProgramManager, dialog := SlPmDialog
KEY54.0 = name := AreaDiagnosis, dialog := SlDgDialog
KEY55.0 = name := AreaStartup, dialog := SlSuDialog
KEY56.0 = name := Custom, dialog := SlEsCustomDialog
```

The area and dialog identifiers can be found in the systemconfiguration.ini from *[System Siemens directory]*\cfg.

# 9.3 Dialog selection via NC

## MMC command in HMI Operate

You can use MMC commands as described in the following:

1. **Definition of MMC commands**
   The following is available in the standard `systemconfiguration.ini` file.
   Combinations:
   address:=MCYCLES --> command:=LM
   address:=CYCLES --> command:=PICTURE_ON
   This distinction is necessary in order to differentiate between measuring cycles and other cycles. This means that:
   - LM is always applicable for measuring cycles, PICTURE_ON, always for non-measuring cycles
   - It is not permissible that newly defined MMC commands are designated PICTURE_ON and LM
2. **"Run MyScreens" license**
   All the dialogs opened by "Run MyScreen", with the exception of measuring cycles, fall under the "Run MyScreens" license, which means that without license, only a limited number of dialogs can be used.
   Example call with test.com (Run MyScreens):

```
g0 f50
MMC("CYCLES,PICTURE_ON,test.com,screen form1","A")
m0
MMC("CYCLES,PICTURE_OFF","N")
M30
```

# Reference lists

# A

## A.1 Lists of start softkeys

### A.1.1 List of start softkeys for turning

#### Program operating area for turning

| HSK1 | HSK2 | HSK3 | HSK4 | HSK5 | HSK6 | HSK7 | HSK8 |
|---|---|---|---|---|---|---|---|
| **Edit** | **Drilling** | **Turning** | **Contour turning** | **Milling** | **Miscellaneous** | **Simulation** | **NC select** |
| HSK9 | HSK10 | HSK11 | HSK12 | HSK13 | HSK14 | HSK15 | HSK16 |
| -- | Straight-line circle (only for ShopTurn) | -- | -- | Measure workpiece | Measure tool | **OEM** | -- |

#### Turning

The following tables list the possible start softkeys for turning technology. Assignments of individual start softkeys can differ depending on the particular system. The specified OEM softkeys are permitted for "Run MyScreens".

programGUIDE (G-Code) start softkeys:

| | **Drilling** | **Turning** | **Contour turning** | | **Milling** | | **Miscellaneous** | |
|---|---|---|---|---|---|---|---|---|
| | **HSK2** | **HSK3** | **HSK4** | | **HSK5** | | **HSK6** | |
| **VSK1** | Centering | Stock removal | Contour | -- | Face milling | Contour | Settings | High speed settings |
| **VSK2** | Drilling reaming | Groove | Stock removal | -- | Pocket | Path | Swivel plane | Parallel axes |
| **VSK3** | Deep-hole drilling | Undercut | Stock removal residual material | -- | Multi-edge spigot | Predrilling | Swivel tool | -- |
| **VSK4** | Boring | Thread | Grooving | -- | Groove | Pocket | -- | -- |
| **VSK5** | Thread | Parting | Grooving residual material | -- | Thread milling | Pocket res. mat. | -- | -- |
| **VSK6** | **OEM** | -- | Plunge-turning | -- | Engraving | Spigot | Subprogram | -- |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **VSK7** | Positions | **OEM** | Plunge turning residual material | **OEM** | **OEM** | Spigot res. mat. | -- | **OEM** |
| **VSK8** | Repeat position. | -- | >> | << | Contour milling | << | >> | << |

ShopTurn start softkeys:

| | **Drilling** | **Turning** | **Contour turning** | | **Milling** | | **Miscellaneous** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **HSK2** | **HSK3** | **HSK4** | | **HSK5** | | **HSK6** | | **HSK10** |
| **VSK1** | Drilling centered | Stock removal | New contour | -- | Face milling | New contour | Settings | High speed settings | Tool |
| **VSK2** | Centering | Groove | Stock removal | -- | Pocket | Path | Swivel plane | Parallel axes | Straight line |
| **VSK3** | Drilling reaming | Undercut | Stock removal residual material | -- | Multi-edge spigot | Predrilling | Swivel tool | Repeat progr. | Circle center point |
| **VSK4** | Deep-hole drilling | Thread | Grooving | -- | Groove | Pocket | Counter-spindle | -- | Circle radius |
| **VSK5** | Thread | Parting | Grooving residual material | -- | Thread milling | Pocket res. mat. | Transformations | -- | Polar |
| **VSK6** | **OEM** | -- | Plunge-turning | -- | Engraving | Spigot | Subprogram | -- | Approach/retract |
| **VSK7** | Positions | **OEM** | Plunge turning residual material | **OEM** | **OEM** | Spigot res. mat. | -- | **OEM** | -- |
| **VSK8** | Repeat position. | -- | >> | << | Contour milling | << | >> | << | -- |

### See also

Defining the start softkey (Page 28)

## A.1.2 List of start softkeys for milling

### Program operating area when milling

| HSK1 | HSK2 | HSK3 | HSK4 | HSK5 | HSK6 | HSK7 | HSK8 |
|---|---|---|---|---|---|---|---|
| **Edit** | **Drilling** | **Milling** | **Contour milling** | **Turning** (only for G code) | **Miscellaneous** | **Simulation** | **NC select** |

| HSK9 | HSK10 | HSK11 | HSK12 | HSK13 | HSK14 | HSK15 | HSK16 |
|---|---|---|---|---|---|---|---|
| -- | Straight-line circle (only for ShopMill) | -- | -- | Measure workpiece | Measure tool | **OEM** | -- |

## Milling

The following tables list the possible start softkeys for milling technology. Assignments of individual start softkeys can differ depending on the particular system. The specified OEM softkeys are permitted for "Run MyScreens".

programGUIDE (G-Code) start softkeys:

| | **Drilling** | **Milling** | **Contour milling** | | **Turning** | | **Miscellaneous** | |
|---|---|---|---|---|---|---|---|---|
| | **HSK2** | **HSK3** | **HSK4** | | **HSK5** | | **HSK6** | |
| **VSK1** | Centering | Face milling | Contour | -- | Stock removal | Contour | Settings | -- |
| **VSK2** | Drilling reaming | Pocket | Path | -- | Groove | Stock removal | Swivel plane | Parallel axes |
| **VSK3** | Deep-hole drilling | Multi-edge spigot | Predrilling | -- | Undercut | Stock removal residual material | Swivel tool | -- |
| **VSK4** | Boring | Groove | Pocket | -- | Thread | Grooving | High speed settings | -- |
| **VSK5** | Thread | Thread milling | Pocket res. mat. | -- | Parting | Grooving residual material | -- | -- |
| **VSK6** | **OEM** | Engraving | Spigot | -- | -- | Plunge-turning | Subprogram | -- |
| **VSK7** | Positions | **OEM** | Spigot res. mat. | **OEM** | **OEM** | Plunge turning residual material | -- | **OEM** |
| **VSK8** | Repeat position. | -- | >> | << | Contour turning | << | >> | << |

ShopMill start softkeys:

| | **Drilling** | **Milling** | **Contour milling** | | **Miscellaneous** | | **Straight line circle** |
|---|---|---|---|---|---|---|---|
| | **HSK2** | **HSK3** | **HSK4** | | **HSK6** | | **HSK10** |
| **VSK1** | Centering | Face milling | New contour | -- | Settings | -- | Tool |
| **VSK2** | Drilling reaming | Pocket | Path | -- | Swivel plane | Parallel axes | Straight line |
| **VSK3** | Deep-hole drilling | Multi-edge spigot | Predrilling | -- | Swivel tool | Repeat progr. | Circle center point |
| **VSK4** | Boring | Groove | Pocket | -- | High speed settings | -- | Circle radius |
| **VSK5** | Thread | Thread milling | Pocket res. mat. | -- | Transformations | -- | Helix |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **VSK6** | **OEM** | Engraving | Spigot | -- | Subprogram | -- | Polar |
| **VSK7** | Positions | **OEM** | Spigot res. mat. | **OEM** | -- | **OEM** | -- |
| **VSK8** | Repeat position. | -- | >> | << | >> | << | -- |

# A.2 List of access levels

The meanings of the different access levels are as follows:

| Protection level | Locked by | Area |
|---|---|---|
| ac0 | Reserved for Siemens | |
| ac1 | Password | Machine manufacturer |
| ac2 | Password | Service |
| ac3 | Password | User |
| ac4 | Keyswitch position 3 | Programmer, machine setter |
| ac5 | Keyswitch position 2 | Qualified operator |
| ac6 | Keyswitch position 1 | Trained operator |
| ac7 | Keyswitch position 0 | Semi-skilled operator |

# A.3 List of colors

## System colors

A uniform color table is available for configuring dialogs (subset of the respective standard colors). The color of an element (text, input field, background, etc.) can be selected from the following options (between 0 and 133).

As an alternative to the predefined colors, you can also specify colors as RGB values ("#RRGGBB").

| Index | Pictogram | Color | Color description |
|---|---|---|---|
| 1 | | Black | |
| 2 | | Orange | |
| 3 | | Dark green | |
| 4 | | Light gray | |
| 5 | | Dark gray | |
| 6 | | Blue | |
| 7 | | Red | |

| Index | Pictogram | Color | Color description |
|---|---|---|---|
| 8 | | Brown | |
| 9 | | Yellow | |
| 10 | | White | |
| 126 | | Black | Font color of an input/output field that is currently in focus |
| 127 | | Light orange | Background color of an input/output field that is currently in focus |
| 128 | | Orange | System color focus |
| 129 | | Light gray | Background color |
| 130 | | Blue | Header color (active) |
| 131 | | Black | Header font color (active) |
| 132 | | Turquoise | Background color of a toggle field |
| 133 | | Light blue | Background color of a list box |

# A.4 List of language codes used in file names

## Supported languages

Standard languages:

| Language | Abbreviation in file name |
|---|---|
| Chinese simplified | chs |
| German | deu |
| English | eng |
| French | fra |
| Italian | ita |
| Spanish | esp |

Other languages:

| Language | Abbreviation in file name |
|---|---|
| Chinese traditional | cht |
| Danish | dan |
| Finnish | fin |

| Language | Abbreviation in file name |
|---|---|
| Indonesian | ind |
| Japanese | jpn |
| Korean | kor |
| Malay | msl |
| Dutch | nld |
| Polish | plk |
| Portuguese (Brazil) | ptb |
| Romanian | rom |
| Russian | rus |
| Swedish | sve |
| Slovakian | sky |
| Slovenian | slv |
| Thai | tha |
| Czech | csy |
| Turkish | trk |
| Hungarian | hun |
| Vietnamese | vit |

## A.5 List of accessible system variables

### References

List Manual System Variables/PGAsl/

## A.6 Behavior when opening the dialog (attribute CB)

The following table provides an overview of the conditions when the CHANGE method is called.

The following applies to attribute CB:

**CB0**

The CHANGE method is triggered when the screen is displayed if the variable has a valid value at this time (e.g. through default setting or NC/PLC variable).

### CB1 (default)

The CHANGE method is not explicitly triggered when the screen is displayed. If the variable has a configured NC/PLC variable, then the CHANGE method is of course still called.

| Condition | | | | Response |
|---|---|---|---|---|
| Type | System or user variable | Default setting | Execute the CHANGE method | |
| I/O field | Yes | Yes | Yes | Due to the configured NC/PLC variable, there is always at least one automatic call of the CHANGE method with the current value of the NC/PLC variable. The pre-assignment of CB does not have an effect. |
| | | | No | |
| | | No | Yes | |
| | | | No | |
| | No | Yes | Yes | The CHANGE method is called with the pre-assigned value. |
| | | | No | The CHANGE method is not called. |
| | | No | Yes | No call because there is no valid value present to call the CHANGE method. |
| | | | No | |
| Toggle | Yes | Yes | Yes | Due to the configured NC/PLC variable, there is always at least one automatic call of the CHANGE method with the current value of the NC/PLC variable. The pre-assignment of CB does not have an effect. |
| | | | No | |
| | | No | Yes | |
| | | | No | |
| | No | Yes | Yes | The CHANGE method is called with the pre-assigned value. |
| | | | No | The CHANGE method is not called. |
| | | No (per default the first value from the toggle is assigned) | Yes | The CHANGE method is called with the first pre-assigned value of the toggle list. |
| | | | No | The CHANGE method is not called. |

## See also

CHANGE (Page 96)

# Tips and tricks

# B

## B.1 General tips

- If possible, use the OPI version for system variables ($ variables).
  Reason:
  Avoids complex name resolution.
  Example:
  Instead of "$R[10]", it is better to use "/Channel/Parameter/R[u1,10]".
- Avoids cyclic (similar) setting, e.g. of the screen and variable HLP property. Compare the value to be set with the current value first, and only set it when it is different.
  Reason:
  Avoid a time-consuming search of the resolution-dependent help screens.
  Example:

```
DEF MyVar=(R3///,"X1",,"mm"/WR1//"$AA_IM[0]")
CHANGE(MyVar)
   IF MyVar.VAL < 100
      HLP="mypic1.png"
   ELSE
      HLP="mypic2.png"
   ENDIF
END_CHANGE
```

**Recommendation:**

```
CHANGE(MyVar)
   IF MyVar.VAL < 100
      IF HLP <> "mypic1.png"
         HLP="mypic1.png"
      ENDIF
   ELSE
      IF HLP <> "smypic2.png"
         HLP="mypic2.png"
      ENDIF
   ENDIF
END_CHANGE
```

- Replace several RNP() functions in succession with one MRNP() function.
  Replace several RDOP() functions in succession with one MRDOP() function.
  Reason:
  Reduces the communication load and increases the performance.
- Do not read drive parameters faster than in a 1 second cycle, slower is better.
  Reason: The communication with the drives can otherwise be extremely disturbed or even cause failures.
- Avoid successive arithmetic operations with system or user variable-connected dialog variables. Use, for example, register (REG[x]) or (invisible) help variables for this.
  Reason: Each assignment of a value results in a write operation in the connected system or user variable.
- Similar code used in different blocks should be combined in a SUB block for reasons of transparency, serviceability and performance (when displaying the screen). It can then be called at the appropriate positions with the CALL() function.
- By monitoring the CPU load in the dialog line (setting in slguiconfig.xml), it is possible to investigate the effects of changes in the screen on the performance.

## B.2 Tips for debugging

- Use the DEBUG() function for diagnostic purposes. When the DEBUG() function is called, the transferred string is written to the easyscreen_log.txt file. The output of information in the dialog line by the DLGL() function can also be helpful.
  However, after completion of the screen development, this function should be removed again for performance reasons or commented out.
- The easyscreen_log.txt file should have no entries after the development of a screen is completed.

## B.3 Tips for the CHANGE method

- Always keep CHANGE methods very short and small, particularly for those whose variables are connected to a system or user variable and change very frequently.
  Reason:
  Increases the screen performance.
- If possible, do not configure any RNP() functions in CHANGE methods. Instead, it is better to configure an invisible variable parallel to the system or user variable to be read and then use this.
  Reason:
  An RNP() function would be issued with each call. Otherwise, the current value which is already available would be simply accessed.

Example:

A name resolution is performed via RNP() for each change of the axis motion in order to read channel-specific machine data:

```
DEF AXIS_POSITION_X =  (R///,""///"$AA_IM[X]")

CHANGE(AXIS_POSITION_X)
      DLGL("Axis """ << RNP("$MC_AXCONF_GEOAX_NAME_TAB[0]") << """
has moved: "
<< AXIS_POSITION_X)
END_CHANGE
```

Keep the channel-specific machine data up-to-date with the aid of an invisible variable, copy each value change to a temporary variable, e.g. register.

This temporary variable can then be used in the CHANGE method of the value change for the axis position without making a name resolution of the machine data each time and the subsequent read access:

```
DEF AXIS_POSITION_X =  (R///,""///"$AA_IM[X]")
DEF AXIS_NAME_X = (S///,""/WR0//"$MC_AXCONF_GEOAX_NAME_TAB[0]")

CHANGE(AXIS_NAME_X)
      REG[0] = AXIS_NAME_X
END_CHANGE

CHANGE(AXIS_POSITION_X1)
   DLGL("Axis """ << REG[0] << """ has moved " << AXIS_POSITION_X)
END_CHANGE
```

- The update rate and therefore the execution of the associated CHANGE method of variables that are connected to system or user variables with very frequent value changes, can be reduced by using the UR variable property, e.g. variable that is coupled to the actual axis values.
  Reason:
  In this way, the associated CHANGE method is executed in a fixed specified grid for a value change.

## B.4 Tips for DO LOOP loops

As loops can impair the performance of SINUMERIK Operate, depending on the configuration, they should be used carefully and if possible without time-intensive actions.

A register (REG[]) should also be used as run variable, because normal display variables (particularly those with OPI connection) also impair the system performance due to the frequent updates or write operations.

Note that the number of script lines executed "in one operation", is currently limited to 10,000. When this number is reached, the script execution is aborted with an appropriate entry in the easyscreen_log.txt file.

Reason:

- Prevention of endless loops
- "Run MyScreens" must remain operable

# Animated elements

C

## C.1 Introduction

### Introduction

The manual describes working with the X3D Viewer with which you can integrate animated and non-animated graphical scenes (animated elements) – called help screens in the following – into the graphical user interface of SINUMERIK Operate as of Version V4.7 SP1 .

You can show motion sequences and parameters in context-sensitive help screens. This helps you to improve the operation of applications and make the user interface more appealing.

Implementation of your initial idea to the final help screen is described in the steps below:

- Creation of graphical elements and 3D models for the help screens that will eventually appear in the X3D Viewer (see Chapter Modeling (Page 212)).
- Creation of the scene description file in which the model data of the graphics file will be assigned to the scenes and animations to be displayed at particular points (see Chapter Structure of the scene description file (Page 219)).
- Conversion of the X3D files and XML files into HMI files (see Chapter Conversion to hmi file (Page 223)).
- C++ integration of the X3D Viewer into your own application (see Chapter Implementation example (Page 225)).
- Application in Run MyScreens (see Chapter Display in Run MyScreens (Page 226)).

### Overview of the file formats

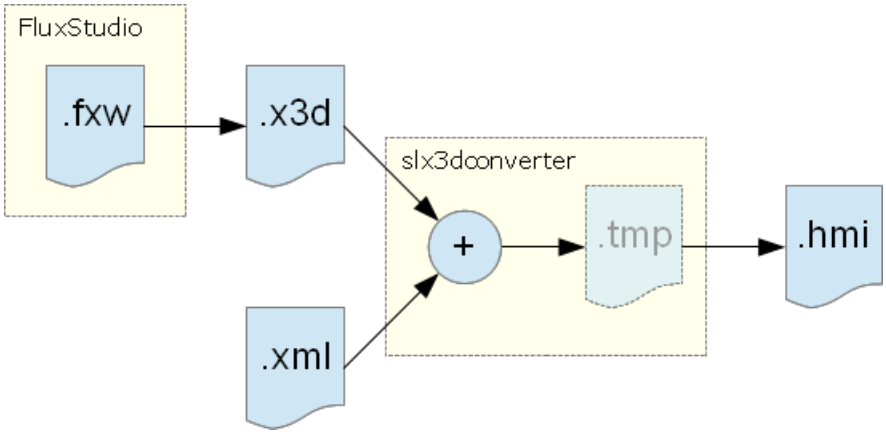



Figure C-1 File formats

| File format | Description |
|---|---|
| .fxw | Flux Studio (3D) graphics file |
| .x3d | Model file in exchange format |
| .xml | Definition file for animations and scenes |
| .hmi | Output file for the X3D Viewer |

# C.2 Modeling

## C.2.1 Requirements

The aim of the modeling is to create a file with the generated 3D models in .x3d file format, an official standard for 3D contents.

Modeling is performed in the **Flux Studio Web3D authoring tool** . The Flux Studio Web3D authoring tool is a freely available 3D modeling tool with versatile export and import options.

### Requirements

- You have installed the Flux Studio Web3D authoring tool, Version 2.1.
- You are familiar with modeling and use of Flux Studio.
- File format .x3d is not described in any detail in this document, adequate knowledge of the subject is assumed.

**Note**

You can download the Flux Studio Web3D authoring tool from the Internet via the following link:

http://mediamachines.wordpress.com/flux-player-and-flux-studio (http://mediamachines.wordpress.com/flux-player-and-flux-studio)

## C.2.2 Rules for modeling

Observe the following rules when modeling. You will only be able to process your help screen file if you have followed these rules correctly.

### Rules for modeling

1. The time sensor must have the name "Animation".
2. All related elements must be assigned to a group.

3. All groups must be set to the following position:
   x = 0,  y = 0,  z = 0
4. To make the groups invisible, the translation values are set as follows:
   x = 1000000,  y = 1000000 and z = 1000000
5. The following elements must have the same name both in the Flux Studio graphic models and in the XML scene definitions (cf. Chapter Overview (Page 218)):
   - Tool
   - Cameras
6. Before you can create a .x3d file you must make the following settings in dialog box "Export Options":

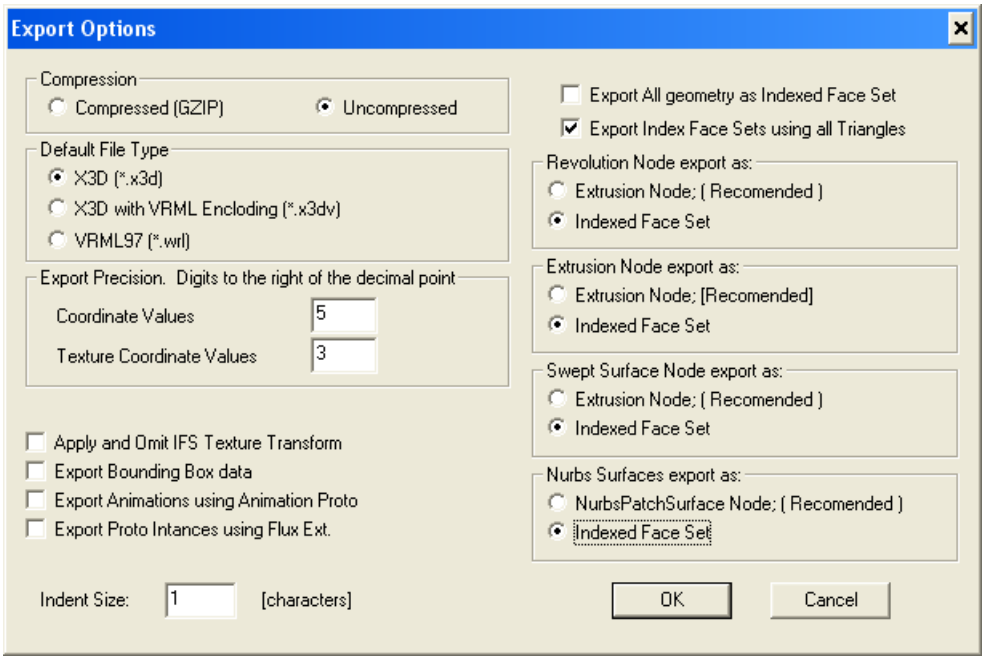



Figure C-2 Settings in the Export Options dialog box

7. We recommend the following settings for texts (see also the following dialog box):
   – Texts must always be centered.
   – The size of the text should be 0.2. This size allows you to position the text easily. The text output by the X3D Viewer is in the font size of the user interface, irrespective of this value.

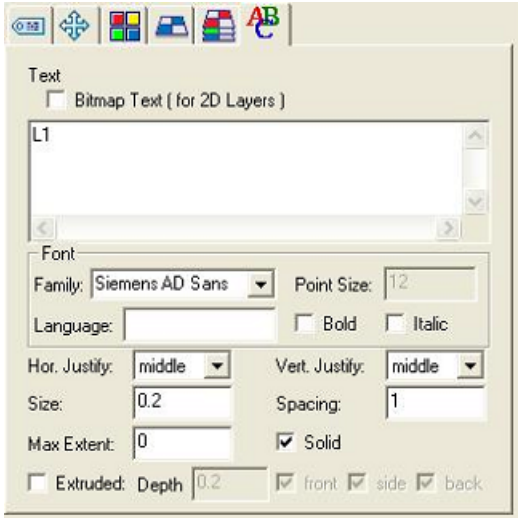



Figure C-3 Settings for text

8. To create hatching, use file schnitt.png as the texture. The lines always go from the bottom left to the top right at an angle of 45°. You should set the scaling to 3 in both dimensions. The rotation depends on the construction type and must be adapted manually.

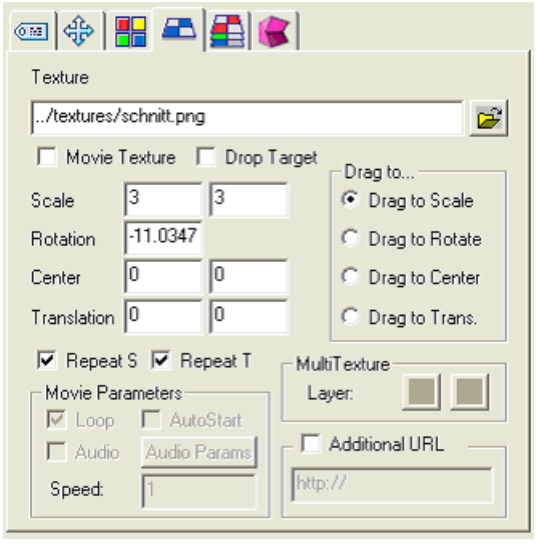



Figure C-4 Settings for hatching

9. We recommend the following sizes/dimensions for the blanks:
   - Cylinder for turning operation:
     Length = 5.5 ; Radius = 1.4 (cf. template *turning_blank.fxw*)
   - Cube for milling operation
     x = 4.75 ; y = 3 ; z = 3 (cf. template *milling_blank.fxw*)

### See also

XML commands (Page 218)

## C.2.3 Importing graphics (models)

In Flux Studio, you can import external graphics (models). You can store frequently used models, such as tools, as .x3d or .hmi files centrally for further use as finished elements in other help screens. You will find a list of supplied modeling templates in Chapter Modeling templates (Page 217).

Complex objects can also be created with other modeling tools, such as Cinema4D, and then imported.

A description of how you can reuse these models is given below.

### Import as inline

Flux Studio uses the object "Inline" for importing .x3d files. It allows you to link in external elements without multiplying the 3D data of these models.

Insert the object with menu item **Create** -> **Create Inline**. Then enter a file name in the object properties.

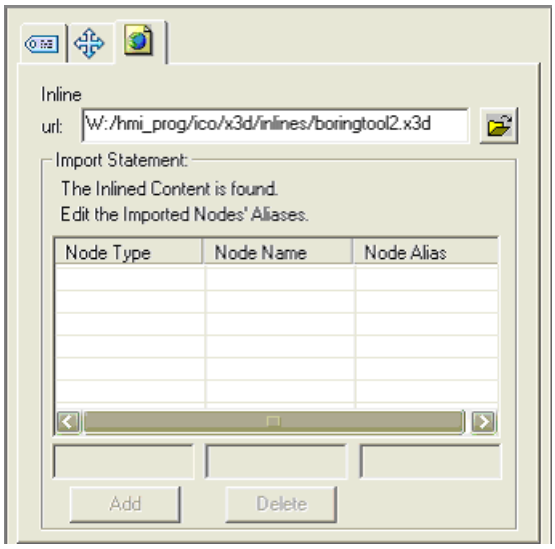



Figure C-5 Importing with object "Inline"

### Importing 3D model data

Proceed as follows to import model data from a file.

1. Open dialog box "Flux Studio Accutrans3D Translation Utility" via menu item **File** -> **Import Other Format** .
2. Select a format from the "File Types" list box. For example, to import a Cinema4D file, select file type "3D Studio."
3. Then select a file with **Select Files** and start import.

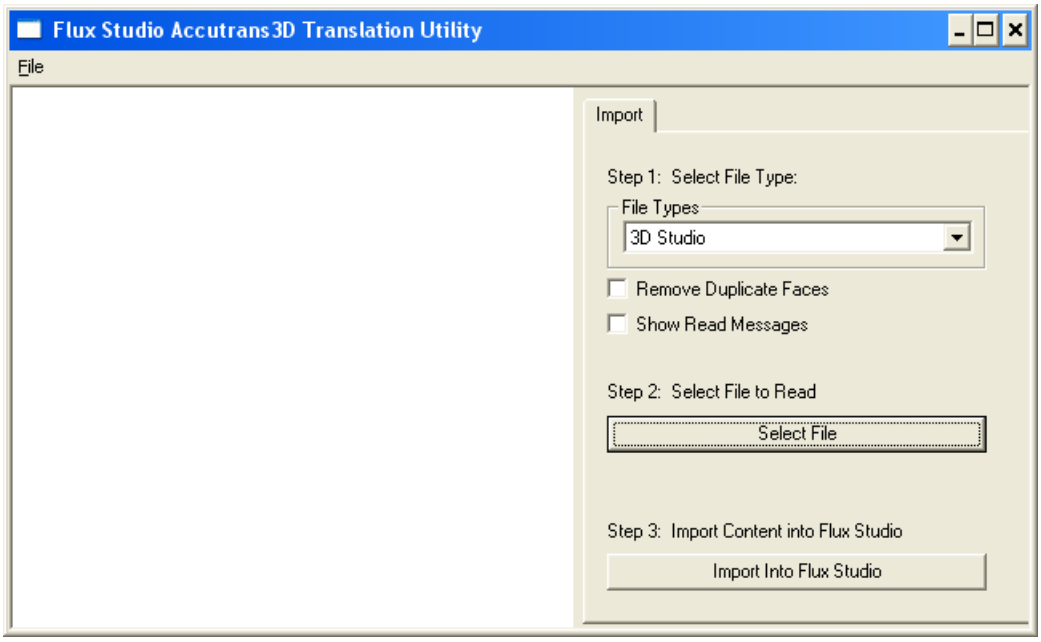



Figure C-6 Importing 3D model data

The model data from the Cinema4D file is inserted as a complete group. All cameras and any other information that is additionally imported should be deleted. Only the graphics elements of the Cinema4D file are required.

## C.2.4 Modeling templates

In this chapter, you will find a list of graphics elements on which you can base your design, colors, and sizes. To maintain a uniform "look and feel" in the user interface, we recommend using the style of the templates or the templates themselves for creating your own graphics.

### General

| Template | Description |
|---|---|
| intersection_texture.png | Texture for an sectioned area |
| rapid_traverse_line_hori.fxw | Horizontal line to illustrate the tool traversing path during rapid traverse |
| rapid_traverse_line_vert.fxw | Vertical line to illustrate the tool traversing path during rapid traverse |

| Template | Description |
|---|---|
| feed_traverse_line_hori.fxw | Horizontal line to illustrate the tool path during feed |
| feed_traverse_line_vert.fxw | Vertical line to illustrate the tool path during feed |
| dimensioning_lines_hori.fxw | Horizontal projection lines |
| dimensioning_lines_vert.fxw | Vertical projection lines |
| z1_inc.fxw | Horizontal dimension line with the identifier Z1 |
| x1_inc.fxw | Vertical dimension line with the identifier X1 |
| 3d_coordinate_origin.fxw | 3D coordinate origin |
| 3d_zero_point.fxw | 3D zero point |

### Turning

| Template | Description |
|---|---|
| turning_blank.fxw | The blank for turning technology: A simple cylinder |
| turning_centerline.fxw | Center line |
| turning_centerpoint.fxw | Coordinate origin to display the zero point in the WCS |
| turning_refpoint.fxw | Reference point, for example, for a machining operation |
| turning_machining_area.fxw | Boundary lines for the machining area |

### Milling

| Template | Description |
|---|---|
| milling_blank.fxw | The blank for milling technology: A simple cube |
| milling_centerline.fxw | Center line |
| milling_refpoint.fxw | Reference point, for example, for a machining operation |

# C.3 XML commands

## C.3.1 Overview

The scene description file (extension .xml) is required so that the X3D Viewer can access the graphics. In the scene description file, you assign the scene names that are called from the configuration to the times in the .x3d file (time sensor) at which the graphics are positioned.

## C.3.2 Structure of the scene description file

The structure of the scene description file is shown below together with an explanation of the relevant XML commands:

### Design and description

**<!-- Treatment of the plane-dependent texts for turning -->**

```
<TextPlane plane="G18_ZXY" />
```

**Description**

| | |
|---|---|
| <TextPlane plane="G18_ZXY" /> | Treatment of the plane-dependent texts (axis names) specifically for turning. If, for example, a text with the identifier "Z" is used (for G18 first geometry axis), the corresponding geometry axis identifier of the control will be shown in the help screen (for example, "Z"). |

**<!--Text alignment -->**

```
<TextPosition center='true' />
```

**Description**

| | |
|---|---|
| <TextPosition center='true' /> | Marking that the texts are centered. This is relevant to the display of the texts in rotated screens. We recommend using this setting. |

**<!--Adapting the tool speed -->**

```
<ToolSpeedFactors planeSpeedFactor="0.4" rapidSpeedFactor="0.7"
reducedSpeedFac¬tor="1.0"/>
```

**Description**

| | |
|---|---|
| <ToolSpeedFactors | This entry can be added if the tool speeds are to be modified. |
| planeSpeedFactor="0.4" | Factor for the feedrate ( 0.4 = 40% ) |
| rapidSpeedFactor="0.7" | Factor for rapid traverse ( 0.7 = 70% ) |
| reducedSpeedFactor="0.1" /> | Factor for a third speed ( 0.1 = 10% ) |

**<!--Definition of an animation-->**

```
<SceneKey name='BoringAnimation' masterRotationSpeed="-64.0"
maxRotAngle="45.0" begin-Time='0.110 endTime='0.118 view='camiso'
speedMaster='boringtool' Type="VIEW_3D_TURN_CYL">
```

**Description**

| | |
|---|---|
| <SceneKey name='BoringAnimation' | Name of the animation |
| masterRotationSpeed="-64.0" | Direction of rotation and rotation speed of the tool.<br>The setting is optional. |
| maxRotAngle="45.0" | To avoid aliasing effects (tool seems to be turning in the wrong direction). The value is usually a fourth of the symmetry of the tool.<br>The setting is optional. |
| beginTime='0.110 | Start time of the animation on the time sensor |
| endTime='0.118' | End time of the animation on the time sensor |
| view='camiso' | Camera to be used for the animation |
| speedMaster='boringtool' | Name of the tool for the animation |
| type="VIEW_3D_TURN_CYL"> | View type defines the view (see Section View type (Page 222)) |

**<!-- Elements -->**

```
<Element name='1' time='0.110' feed='rapid' dwell='2'/>
<Element name='2' time='0.112' feed='rapid'/>
<Element name='3' time='0.114' feed='rapid'/>
<Element name='4' time='0.116' feed='plane'/>
<Element name='5' time='0.118' feed='plane'/>
```

**Description**

| | |
|---|---|
| <Element name='1' | Specifies that it is the first element of the animation. |
| time='0.110' | Time of the first element on the time sensor |
| feed='rapid' | Traversing speed of the element<br>plane = machining feedrate<br>rapid = rapid traverse<br>reduced = reduced speed, slower than "plane" |
| dwell='2'/> | Pause in the traversing motion of the animation. A rotating tool continues to rotate. |

**<!-- End of animation definition -->**

```
</SceneKey>
```

**Description**

| | |
|---|---|
| </SceneKey> | End of the animation definition |

**<!-- Existing animation as source for a new animation -->**

```
<SceneKey source=''BoringAnimation' name=''BoringAnimationRear' mirror='
MirrorScreenX ' />
```

**Description**

| | |
|---|---|
| <SceneKey source="BoringAnimation' | Name of an animation that is to use another animation as its source. |
| name="BoringAnimationRear' | Name of the new animation based on the source |
| mirror=' MirrorScreenX ' /> | Mirroring in the direction of the X axis, output is in the new animation. |

**<!-- Definition of a static scene (screen), (four examples) -->**

```
<SceneKey name='Default' time='0.026' view='camiso'
type="VIEW_3D_DRILL_CUT"/>
<SceneKey name='Cut' time='0.046' view='camside' type="VIEW_SIDE"/>
<SceneKey name='Z1ink' time='0.056' view='camside' type="VIEW_SIDE"
 highLightedGroup='dad_Z1ink'/>
<SceneKey name='Z1abs' time='0.066' view='camside' type="VIEW_SIDE"
 highLightedGroup='dad_Z1abs'/>
```

**Description**

| | |
|---|---|
| <SceneKeyname=’Z1abs' | Screen name |
| time='0.066' | Time of the screen on the time sensor |
| view='camside' | Camera to be used for the screen |
| type="VIEW_SIDE" | View type defines the view (see Section View type (Page 222)) |
| highLightedGroup='dad_Z1abs'/> | Name of the group to be highlighted. The group has been called "Z1abs" in Flux Studio. The prefix "dad_" is automatically added by Flux. |

**<!-- End of xml file –!>**

## C.3.3 Mirroring and rotations

Mirroring or rotation of the image or model is defined with the **mirror** command.

### Description

| | |
|---|---|
| mirror="RotateScreenX" | The screen is rotated around the X axis |
| mirror="RotateScreenY" | The screen is rotated around the Y axis |
| mirror="RotateScreenZ" | The screen is rotated around the Z axis |
| mirror="MirrorScreenX” | The screen is mirrored along the X axis |
| mirror="MirrorScreenY" | The screen is mirrored along the Y axis |
| mirror="MirrorScreenZ" | The screen is mirrored along the Z axis |
| mirror="RotatePieceX” | The model is rotated around the X axis |
| mirror="RotatePieceY" | The model is rotated around the Y axis |
| mirror="RotatePieceZ" | The model is rotated around the Z axis |
| mirror="MirrorPieceX" | The model is mirrored along the X axis |
| mirror="MirrorPieceY" | The model is mirrored along the Y axis |
| mirror="MirrorPieceZ" | The model is mirrored along the Z axis |

All rotate and mirror options can be combined. A rotation angle must be defined for rotation.

### Examples

mirror="RotatePieceZ=180"

mirror="RotatePieceZ=-90 MirrorScreenX"

mirror="RotateScreenY=90"

mirror="MirrorPieceX MirrorPieceY"

mirror="MirrorPieceZ RotatePieceX=-90"

## C.3.4 View type

You define the views with the view type. The views are based on empirical values and rules that produce a sensible representation of the objects.
This ensures that the representation of the objects matches the coordinate system settings made in the user interface (MD 52000 $MCS_DISP_COORDINATE_SYSTEM or MD 52001 $MCS_DISP_COORDINATE_SYSTEM_2).

### Description

| | |
|---|---|
| "VIEW_STATIC" | Direct view without conversion |
| "VIEW_3D_TURN_CYL" | 3D view for turning operations (cylinder) |
| "VIEW_3D_MILL_CUBE" | 3D view for milling operations (cube) |
| "VIEW_3D_DRILL_CUT" | 3D view for drilling operations (sectional view) |

| | |
|---|---|
| "VIEW_SIDE" | 2D view for drilling operations (sectional view) |
| "VIEW_TOP_GEO_AX_1" | 2D view from the direction of the 1st geometry axis |
| "VIEW_TOP_GEO_AX_2" | 2D view from the direction of the 2nd geometry axis |
| "VIEW_TOP_GEO_AX_3" | 2D view from the direction of the 3rd geometry axis |

# C.4 Conversion to hmi file

**Note**

The X3D files and their associated XML files are converted to HMI files when the HMI is powered up.

For each X3D file, a corresponding XML file of the same name must be created. To do this, you store the X3D files and XML files in directory `search path of HMI\ico\x3d \turning` or `milling`. You proceed in the same way with the .ts files.

# C.5 Display in Create MyHMI /3GL

## C.5.1 X3D Viewer

If you want to display particular help screens in a separate OA application you must integrate the X3D Viewer widget into its own OA application.

The X3D Viewer widget provides interfaces, which enable X3D contents to be presented in the HMI.

Class SlX3dViewerWidget is provided for displaying graphical scenes. The definition of the class can be found in the corresponding slx3dviewerwidget.h header file in the global GUI include directory `\hmi_prog\gui\include`.

## C.5.2 Class SlX3dViewerWidget

The class provides a flexible widget that autonomously displays the contents of a model file specified during runtime and, if required, runs the animation.

The interface of the class comprises the Constructor, Destructor, and two methods for controlling the graphical output.

As a direct derivative of the Qt class QWidget, a much more expansive interface is available, which will not be described in any further detail here, (for example, show(), hide() and resize(...). For more information, please refer to the Qt documentation).

## C.5.3 Public methods

### SlX3dViewerWidget ( QWidget* pParent = 0 )

Constructor of the X3D Viewer widget.

| Parameters | Meaning |
|---|---|
| pParent | The parameter is passed on to the constructor of the Qwidget. |

### ~SlX3dViewerWidget ( )

Destructor of the X3D Viewer widget.

| Parameters | Meaning |
|---|---|
| - | - |

## C.5.4 Public slots

### void viewSceneSlot ( const QString& rsFileName, const QString& rsScene, const QString& rsAnimationScene, int nChannel, int nPlane, SlStepTechnology nTechnology )

With the viewSceneSlot method, the X3D Viewer is instructed to load the rsScene static scene and the rsAnimationSzene animated scene from the rsFileName file and display them alternately.

The static scene and then the animated scene are repeatedly alternately displayed for a fixed time.

If no static scene is specified, the animation will be shown immediately; an animated scene may also not have been specified.

Channel number, plane and technology are used to rotate the scenes to the correct position (depending on the set machine coordinate system).

| Parameters | Meaning |
|---|---|
| rsFileName | Name of the file that contains the scenes to be displayed |
| rsScene | Name of the static scene. |
| rsAnimationScene | Name of the animated scene. |
| nChannel | Channel number |

| Parameters | Meaning |
|---|---|
| nPlane | Plane |
| nTechnology | Technology<br>The following constants are defined for the SlStep-Technology enumeration type:<br>• SL_STEP_NO_TECHNOLOGY<br>• SL_STEP_MILLING<br>• SL_STEP_TURNING<br>• SL_STEP_SURFACE_GRINDING<br>• SL_STEP_CIRCULAR_GRINDING |

### void viewSceneSlot ( const QString& rsFileName, const QString& rsScene, const QString& rsAnimationScene )

A simplified form of the viewSceneSlot method with which the X3D Viewer can be instructed to load and display the rsScene static scene and/or the rsAnimationSzene animated scene from the rsFileName file.

| Parameters | Meaning |
|---|---|
| rsFileName | Name of the file that contains the scenes to be displayed. |
| rsScene | Name of the static scene. |
| rsAnimationScene | Name of the animated scene. |

## C.5.5 Libraries

In order to be able to apply the X3D Viewer in your own projects, the list of library dependencies must be extended by the 'slx3dviewer.lib' entry.

## C.5.6 Implementation example

You will find an implementation example in the package Create MyHMI/3GL under **\examples \GUIFrameWork\SlExGuiX3D**.

## C.5.7 Machine data

Display MD 9104 $MM_ANIMATION_TIME_DELAY

Time delay in seconds until the animation is started for help screens. The setting does not apply to help screens that are only animated.

The setting applies globally to all animations of SINUMERIK Operate.

### C.5.8 Notes about use

- The animation is interrupted if the X3D Viewer widget is hidden. If this happens, there is no need to choose a scene selection without animation.
- You should avoid frequent or multiple instantiation of the X3D Viewer widget as this reduces performance and memory capacity. The use (implementation) of an X3D Viewer Singleton is recommended in such application scenarios.
- The X3D Viewer can also be used in signal slot concepts.
- If an error occurs (for example, a file not found or unknown scene name), the X3D Viewer widget opens a message area. This area is automatically hidden again when another help screen is activated.

## C.6 Display in Run MyScreens

This section is intended for experienced "Run MyScreens" developers. Necessary background knowledge is provided in the associated documentation.

In addition to using screens in .bmp or .png format, animated help screens can also be displayed with the X3D Viewer.

To integrate help screens, use the Run MyScreens interface in the usual way.

To output images in .bmp or .png format, XG0 is entered in the definition of a dialog box in the Attributes section or the parameter is left empty. To integrate X3D help screens in a dialog, you must enter **XG1** in the attributes.

```
//M(MASK_F_DR_O1_MAIN/$85407////52,80/XG1)
```

The following parameters are required to activate the individual help screens; their meaning is described in Section 5.3 Public slots:

1. File name
2. StaticScene (optional)
3. AnimationScene (optional)
4. Technology (optional)
5. Plane (optional)

The parameters are combined into a character string in this order, separated by a comma.

```
Hlp = "File name,StaticScene,AnimationScene,Technology,Plane"
```

## Examples

- The default help screen can be set with the specified variable Hlp.
  In the following example, the "MyAnimation" animation is output from the "MyDlgHelp.hmi" file. No StaticScene is specified, i.e. no static scene is output. No specifications are made for the Technology and Plane, i. e. default values are used.
  ```
  Hlp = "MyDlgHelp.hmi,,MyAnimation"
  ```
- The hlp property can be set on a specific help screen for the individual parameters of the input screen.
  In the following example, the static "MyParam" scene and the "MyAnimation" animation are output from the "MyDlgHelp.hmi" file in the G17 plane. No specification is made for the Technology, i.e. a default value is used.
  ```
  VarMyParam.hlp = "MyDlgHelp.hmi,MyParam,MyAnimation,,G17"
  ```

# Glossary

# 10

### Action

Anything configured within a → Method: → Functions, → Calculating variables, → Changing properties, etc.

### User variable

Variables defined by the user in the → Parts program or data block.

### Array

An array can be used to organize data of a standard data type stored in the memory in such a way that it is possible to access the data via an index.

### Attribute

Characteristic that assigns specific → Properties to an object (→ Dialog or → Variable).

### Menu tree

A group of interlinked → Dialogs

### Group

Reload unit for → Configuration file

### Dialog

Display of the → User interface

- **Dialog-dependent softkey menu**
  Softkey menu, which is called from a newly configured dialog.
- **Dialog-independent softkeys**
  Softkeys, which are not called from a dialog, i.e., start softkey and softkey menus, which the user configures before the first, new dialog.

### Definition lines

Program section in which → Variables and softkeys are defined

### Editor

ASCII Editor with which characters can be entered in a file and edited.

### Properties

Characteristics of an object (e.g of a → Variable)

### Input/output field

Also I/O field: for inputting or outputting variable values.

### Start softkey

Softkey with which the first newly created → Dialog is started.

### Event

Any action, which initiates execution of a → Method: Input of characters, actuation of softkeys, etc.

### Help variable

Internal arithmetic variable to which no → Properties can be assigned and is not, therefore, visible in the → Dialog.

### Hotkeys

6 keys on OP 010, OP 010C and SINUMERIK keyboards with hotkey blocks. Pressing the keys selects an operating area directly. As an option, 2 additional keys can be configured as hotkeys.

### Interpreter

The interpreter automatically converts the defined code from the → Configuration file into a → Dialog and controls its use.

### Method

Programmed sequence of operations executed when a corresponding → Event occurs.

### Parameter

Parameters are variable elements of the programming syntax and are replaced by other words/ symbols in the → Configuration file.

### PI service

Function which, on an NC, executes a clearly defined operation. PI services can be called from the PLC and the HMI system.

### PLC hard key

PLC hard keys are provided via the PLC interface of the HMI software, just like hotkeys. The functions, which can be triggered via the user interface, can be configured.

They take the form of MCP keys or evaluations of PLC signal logic operations in the PLC user program. For this reason, they are referred to as "virtual keys".

### Programming support

Provision of → Dialogs to assist programmers in writing → Parts programs with "higher-level" components

### Configuration file

File, which contains definitions and instructions that determine the appearance of → Dialogs and their → Functions.

### Recompile

NC code sections can be generated in a → Part program from input fields in → Dialogs in the → Programming support system. Recompilation is the reverse operation. The input fields used to generate a selected section of NC code are retrieved from the NC code and displayed in the original dialog.

### Simulation

Simulation of a → Parts program run without movement of the actual machine axes.

### Softkey labels

Text/image on the screen, which is assigned to a softkey.

### Softkey menu

All horizontal or all vertical softkeys

### Column index

Column number of an array

### System variable

NC variable providing the NC states for parts program accesses and accesses from HMI.

### Selecting

A program formulated in the NC language, which specifies motion sequences for axes and various special actions.

### Toggle field

A list of values in the → Input/output field; check with toggle field: The value input in a field must be the same as one of the listed values.

### Variable

Designation of a memory location, which can be displayed in a → Dialog by assigning → Properties and in which input data and the results of arithmetic operations can be entered.

### Line index

Row number of an array

### Access level

Graduated system of authorization, which makes the accessibility and utilization of functions on the operator interface dependent on the authorization rights of the user.

# Index


## A

Access level, 52
Alarms
  Language code, 203
Array
  Access mode, 167
  Column index, 167
  Compare mode, 167
  Definition, 165
  Element, 166
  Line index, 167
  Status, 170
Attributes, 71

## B

Background color, 73

## C

Colors, 73
Comparison operators, 93
Conditions, 94
Configuration file, 25
Configuration syntax"; "Extended syntax, 36
Configuration syntax"; "Screens, 36
Configuration syntax"; "Softkey, 37
Configuration syntax"; "Table columns, 37
Configuration syntax"; "Variables, 36
Configuring PLC softkeys, 193
Constants, 93
Custom widget
  Automatic data exchange, 178
  Definition, 175
  Executing a method, 180
  Interface, 176
  Library, 175
  Manual data exchange, 179
  Reading and writing properties, 179
  Responding to a custom widget signal, 182

## D

Default setting, 70
Defines the softkey menu, 51
Dialog
  Definition, 39
  Definition block, 40
  Multiple columns, 47
  Properties, 41
Dialog change mode, 135
Dialog element, 45
Display mode, 71
Display option, 71
DLL file, 127

## E

ELLIPSE (define circle), 164
ELLIPSE (define ellipse), 164

## F

File
  Copy, 114
  Delete, 114
  Moving, 116
File formats
  fxw, 211
  hmi, 211
  x3d, 211
  xml, 211
Flux Studio Web3D authoring tool, 212
Focus control, 174
Foreground color, 73
Function
  CALL (Subprogram call), 110
  CLEAR_BACKGROUND), 113
  CP (Copy Program), 114
  CVAR (Check Variable), 112
  Cyclic execution of scripts, 160
  DEBUG, 121
  DLGL (Dialog line), 120
  DO LOOP), 158
  DP (Delete Program), 114
  EP (Exist Program), 115
  EVAL (Evaluate), 126
  EXIT, 122
  EXITLS (EXIT Loading Softkey), 127
  FCT, 127
  File access, 118
  FORMAT (string), 156
  GC (Generate Code), 129

HMI_LOGIN, 132
HMI_LOGOFF, 132
HMI_SETPASSWD, 132
INSTR (String), 152
LA (Load Array), 133
LB (Load Block), 134
LEFT (string), 153
LEN (string), 152
LISTADDITEM), 124
LISTCLEAR), 124
LISTCOUNT), 124
LISTDELETEITEM, 124
LISTINSERTITEM, 124
LM (Load Mask), 135
LS (Load Softkey), 136
MIDS (string), 154
MP (Move Program), 116
MRDOP), 109
MRNP (Multiple Read NC PLC), 139
Overview, 108
PI_START, 141
RDFILE), 118
RDLINEFILE), 118
RDOP), 109
Reading and writing drive parameters, 109
Recompile NC code, 146
Recompile without comment, 148
REPLACE (string), 154
RESIZE_VAR_IO, 143
RESIZE_VAR_TXT, 143
RETURN (Back), 145
RIGHT (string), 153
RNP (Read NC PLC Variable), 142
SB (Search Backward), 150
SF (Search Forward), 150
SP (Select Program), 117
START_TIMER), 160
STOP_TIMER, 160
STRCMP (string), 155
STRINSERT (string), 155
STRREMOVE (string), 155
SWITCH), 139
TRIMLEFT (string), 156
TRIMRIGHT (string), 156
UNTIL, 158
WDOP, 109
WHILE, 158
WNP (Write NC PLC Variable), 142
WRFILE, 118
WRLINEFILE), 118

## G

Generates NC code, 129
Graphic text, 70
Grid → Table, 170

## H

H_SEPARATOR (define horizontal dividing line), 165
Help display, 73
Help file, 74
Help variable, 62

## I

I/O field with integrated unit selection, 74
Image as short text, 66
Import
  Graphics (models), 215
Input mode, 72

## L

Language code, 203
Limits, 70
LINE (define line), 163
List field, 66
Long text, 70

## M

Machine data, 225
Machining step support, 148
Master dialog, 135
Mathematical functions, 92
Menu tree, 27
Method
  ACCESSLEVEL, 96
  CHANGE, 96
  CHANNEL, 98
  CONTROL, 98
  LANGUAGE, 100
  LOAD, 100
  LOAD GRID, 137
  OUTPUT, 102
  Overview, 96
  PRESS, 103
  PRESS(ENTER), 104

PRESS(TOGGLE), 104
RESOLUTION, 105
RESUME, 105
SUSPEND, 106
UNLOAD, 101

## N

NC variable
Read, 142
Write, 142
Numerical format, 77

## O

Online help, 58
Operator
Bit, 94
Mathematical, 91

## P

Password dialogs, 48
Password input mode (asterisk), 69
PI services, 108
PLC variable
Read, 142
Write, 142
Position
Input/output field, 73, 80
Short text, 73, 80
Progress bar with two color changes, 67
Progress bar without color change, 68
Protection levels, 202
Public slots, 224

## R

RECT (defining a rectangle), 163
Registers
Exchanging data, 144
Status, 145
Value, 144

## S

Short text, 70
Signal color"; "progress bar, 73
Signal values"; "progress bar, 70
SIX3dViewerWidget, 223
Softkey
Assign properties, 51
Properties, 53
Start softkey, 27, 28
Strings, 80
Sub-dialog, 135
Subprogram, 108
Block identifier, 111
Call, 110
cancel, 145
Variable, 111
System colors, 202
System variable, 63, 73

## T

Table
Defining columns, 173
Definition, 170
Programming, 172
Text, 70
Toggle field, 65, 70, 78
Toggle symbol, 71
Tooltip, 70, 72
Trigonometric functions, 92

## U

Unit text, 70
Update rate, 71
User variable, 73

## V

V_SEPARATOR (define vertical dividing line), 164
Variable
calculating, 62
Change property, 74
Check, 112
CURPOS, 82
CURVER, 82
End, 122
ENTRY, 83
ERR, 83
FILE_ERR, 84
FOC, 85
Parameter, 69
S_ALEVEL, 86
S_CHAN, 87
S_CONTROL, 87
S_LANG, 88

S_NCCODEREADONLY, 89
S_RESX, 89
S_RESY, 89
Variable status, 61
Variable type, 69
INTEGER, 75
VARIANT, 76
Variable value, 61

## W

Write mode, 73

# Appendix

# A

## A.1 Documentation overview SINUMERIK 828D

