SIEMENS

SINUMERIK

SINUMERIK 808D Function Manual

Operating Instructions

Valid for:
SINUMERIK 808D Turning (software version: V4.4.2)
SINUMERIK 808D Milling (software version: V4.4.2)

Target group:
Project engineers, technologists (from machine manufacturers), system startup engineers (systems/machines), and programmers

12/2012
6FC5397-2EP10-0BA0

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: 6FC5397-2EP10-0BA0
℗ 12/2012 Technical data subject to change

Preface

SINUMERIK 808D documentation

The SINUMERIK 808D documentation consists of the following components:

- Operating Instructions
  - Mechanical Installation Manual
  - Electrical Installation Manual
  - PLC Subroutines Manual
  - Function Manual
  - Parameter Manual
- Diagnostics Manual
- Commissioning Manual
- Programming and Operating Manual (Turning)
- Programming and Operating Manual (Milling)
- Manual Machine Plus (Turning)
- Online Help for Programming and Operating (Turning)
- Online Help for Programming and Operating (Milling)
- Online Help for Manual Machine Plus (Turning)

My Documentation Manager (MDM)

Under the following link you will find information to individually compile your documentation based on the Siemens content:

www.siemens.com/mdm

Target group

This manual is intended for use by project engineers, technologists (from machine manufacturers), system start-up engineers (control systems/machines), and programmers.

Benefits

This manual provides the target group with the information required for implementing the desired functions.

Standard scope

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

Other functions not described in this documentation might be executable in the control. However, no claim can be made regarding the availability of these functions when the equipment is first supplied or in the event of servicing.

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

Hotline:	+86 400-810-4288
Service and Support	• China: www.siemens.com.cn/808D • Worldwide: http://support.automation.siemens.com

EC Declaration of Conformity

The EC Declaration of Conformity for the EMC Directive can be found on the Internet at http://support.automation.siemens.com

Here, enter the number **15257461** as the search term or contact your local Siemens office.

Licensing provisions

The SINUMERIK 808D software is protected by national and international copyright laws and agreements. Unauthorized reproduction and distribution of this software or parts thereof is liable to prosecution. It will be prosecuted both according to criminal and civil law and may result in severe punishment or demands for compensation.

In the SINUMERIK 808D software, open source software is used. The licensing provisions for this software are included on the Toolbox DVD and are to be observed accordingly.

Table of contents

Introduction

1

Notations

The following notation and abbreviations are used in this documentation:

- Programmable logic control (PLC) interface signals -> IS "Signal name" (signal data)

  Example: IS "Feedrate override" (DB380x.DBB0)

  The variable byte is located in the "to axis" range, x stands for the axis:

  0 Axis 1

  1 Axis 2

  n Axis n+1.

- Machine data -> MD MD_NR MD_NAME (description)

  e.g.: MD30300 IS_ROT_AX (rotary axis)

- Setting data -> SD SD_NR SD_NAME (description)

  e.g.: SD41200 JOG_SPIND_SET_VELO (JOG velocity for the spindle)

- The chapter titles are supplemented by a code in brackets (e.g. Chapter 1: EMERGENCY STOP (N2)). This brief description is used in cross references to other chapters.

The machine and setting data are divided into the following areas:

Range	Data area	Meaning
200 - 9999	$MM_	Display machine data
10,000 - 19,999	$MN_	General machine data
20,000 - 28,999	$MC_	Channel-specific machine data
30,000 - 38,999	$MA_	Axis-specific machine data
41,000 - 41,999	$SN_	General setting data
42,000 - 42,999	$SC_	Channel-specific setting data
43,000 - 43,999	$SA_	Axis-specific setting data

Explanations for the technical data

Data types:
The following data types are used in the control:

- DOUBLE
  Floating-point value (64-bit value)
  Input limits from +/-4.19*10^{-307} to +/-1.67*10^{308}
- DWORD
  Integer values (32-bit values)
  Input limits from -2,147,483,648 to +2,147,483,648 (decimal);
  as hexadecimal value: 0000 through FFFF
- BYTE
  Integer values (8-bit values)
  Input limits from -128 to +127 (decimal); as hexadecimal value: 00 through FF
- BOOLEAN
  Boolean value: TRUE (1) or FALSE (0)
- STRING
  Consisting of max. 16 American Standard Code for Information Interchange (ASCII) characters (upper-case letters, numbers and underscore)

Detailed explanations

- Detailed explanations for the machine/setting data and interface signals used can be found in the SINUMERIK 808D Parameter Manual.
- Detailed explanations of the alarms which may occur can be found in the SINUMERIK 808D Diagnostics Manual.

See also

Various Interface Signals (Page 13)

Various Interface Signals

2

2.1 General

Brief description

This chapter describes the functionality of various interface signals which are of general relevance, but are not described in the function-specific chapters.

Interfaces

The exchange of signals and data between the PLC user program and the NCK (numerical control kernel) or HMI (display unit) is performed via various data areas. The PLC user program does not have to handle the exchange of data and signals. From the user's point of view, this takes place automatically.

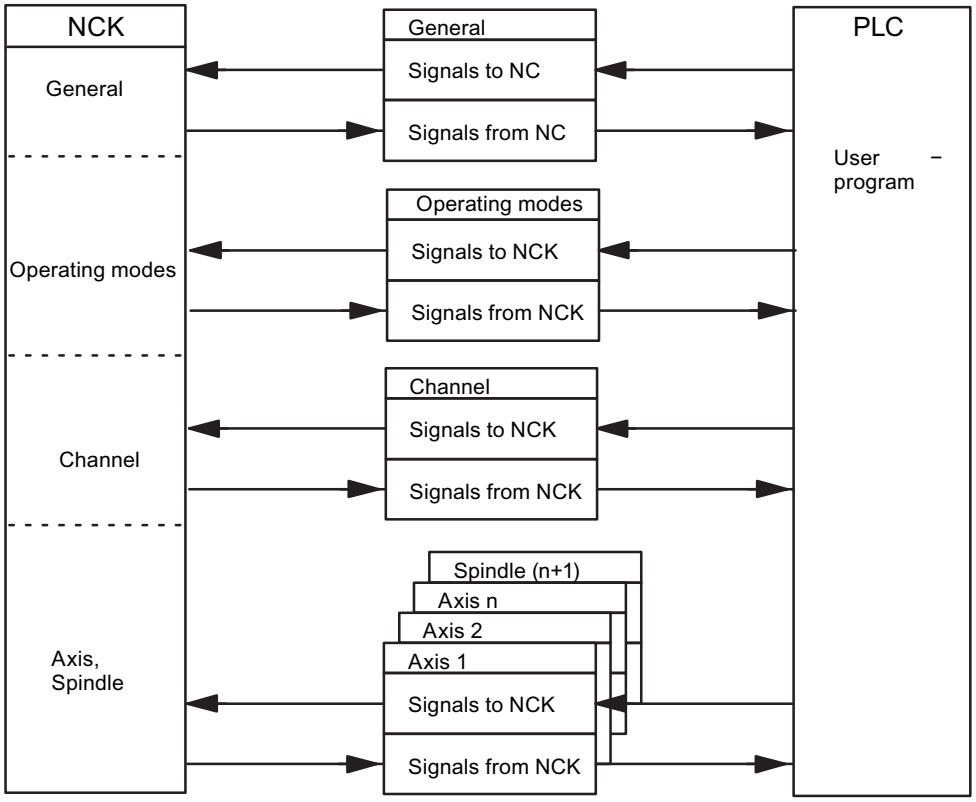


Figure 2-1 PLC/NCK interface

Cyclic signal exchange

The control and status signals of the PLC/NCK interface are updated cyclically.

The signals can be subdivided into the following groups (see previous figure):

- General signals
- Operating mode signals
- Channel signals
- Axis/spindle signals

2.2 Signals from PLC to NCK

2.2.1 Access authorization

Access authorization

Access to programs, data, and functions is user-oriented and controlled via protection levels. The SINUMERIK 808D provides a concept of access levels for enabling data areas. You can view such information from the table below:

Access level	Default password	Target group
Manufacturer (level 2)	SUNRISE	OEMs
Customer (level 3)	CUSTOMER	End users
No password	-	-

This provides a multi-level safety concept for controlling access rights.

Reference:

Commissioning Manual, Section: Access levels

2.2.2 General signals

Delete distance-to-go (DB3200.DBX6.2)

IS "Delete distance-to-go (channel specific)" is only active for path axes.

With the rising edge of the interface signal, the distances-to-go of all axes in the geometry grouping are deleted and thus brought to a standstill with ramp stop. The next program block is then started.

Axis/spindle disable (DB380x.DBX1.3)

IS "Axis/spindle disable" can be used for test purposes.

Axis disable (for axis):

If IS "Axis disable" is output - for this axis - no more position partial setpoints are output to the position controller; the axis travel is therefore disabled. The position control loop remains closed and the remaining following error is reduced to zero. If an axis is moved with axis disable the actual value position display shows the setpoint position and the actual velocity value display shows the setpoint velocity even though the machine axis is not actually moving. IS "RESET" (DB3000.DBX0.7) sets the position actual value display to the real actual value of the machine. Travel commands continue to be output to the PLC for this axis. If the interface signal is cancelled again the associated axis can again traverse normally. If the interface signal "Axis disable" is set for a traversing axis, the axis is stopped with a ramp stop.

Spindle disable (for spindle):

If IS "Spindle disable" is set, no more speed setpoints are output to the speed controller in openloop control mode and no more position partial setpoints are output to the position controller in positioning mode. The movement of the spindle is thus disabled. The speed actual value display displays the speed setpoint value. The spindle disable is cancelled via "Reset" or program end (M2) and program restart. If interface signal "Spindle disable" is set while a spindle is turning, the spindle is stopped according to its acceleration characteristic.

Deactivation:

Cancellation of the "Axis/spindle disable" (edge change 1 → 0) does not take effect until the axis/spindle is stationary (i.e. an interpolation setpoint is no longer present). The new movement begins with new specified setpoints. (E.g. new program block with movement specifications in the "AUTO" operating mode.)

Note: actual values vary between simulated and real axis!

Follow-up mode (DB380x.DBX1.4)

If an axis/spindle is operating in follow-up mode, then its setpoint position is made to track the current actual value position. The position setpoint in follow-up mode is not defined by the interpolator but derived from the actual position value. Since recording of the actual position value of the axis continues, it is not necessary to re-home the axis when follow-up mode is cancelled.

Standstill, clamping and positioning monitoring are not effective in follow-up mode.

Effect:
The IS "Follow-up mode" is only of relevance if the drive controller enable has been removed (e.g. by IS "Controller enable" = 0 signal or because of a fault in the control system), or because controller enable is being re-issued.

IS "Follow-up mode" = 1:
If "Controller enable" is removed the position setpoint of the relevant axis is continuously corrected to the actual value. This state is signaled to the PLC by means of IS "Follow-up mode active" (DB390x.DBX1.3). If the "Controller enable" is enabled again and a part program is active, a control internal re-positioning operation is initiated (REPOSA: linear approach with all axes) to the last programmed position. Otherwise, the axis movement starts at the new actual position (which may have changed).

IS "Follow-up mode" = 0:
If "Controller enable" is removed, the old position setpoint is maintained. If the axis is pushed out of position, a following error between position setpoint and actual value results which is corrected when IS "Controller enable" is set. The axis movement starts from the setpoint position valid before the "controller enable" was removed. IS "Followup mode active" (DB390x.DBX1.3) is set to 0 signal during the "Hold" state. Clamping or standstill monitoring is active.

Position measuring system 1 (DB380x.DBX1.5)

A position measuring system may be connected to the spindle. In this case the signal for the spindle has to be set.

Axes always require this signal. In this case, a position measuring system must be installed.

Controller enable (DB380x.DBX2.1)

When the controller enable is activated for the drive, the position control loop of the axis/spindle is closed. The axis/spindle is then subject to position control.

When the controller enable is removed the position control loop and, with a delay, the speed control loop of the axis/spindle are opened.

IS "Position controller active" (DB390x.DBX1.5) is set to 0 signal (checkback).

Activation:

The controller enable for the drive can be set and removed from the following places:

1. From the PLC user program with interface signal "Controller enable" (in normal cases)

   Application: Removal of controller enable before clamping an axis/spindle.

2. The controller enable is cancelled internally by the control when certain faults occur in the machine, the drive, the position measuring system or the control (when faults occur)

   Application: The traversing axes must be brought to a standstill by a rapid stop due to a fault.

3. By the control if the following event occurs: IS "EMERGENCY STOP" (DB2600.DBX0.1) is active

Removal of controller enable for a moving axis/spindle:

- The spindle is braked to standstill with rapid stop taking account of MD36610 AX_EMERGENCY_STOP_TIME (duration of the braking ramp in error states). Alarm 21612 "Controller enable reset during movement" is then triggered.
- The position control loop of the axis/spindle is opened. Checkback signal to PLC with IS "Position controller active" (DB390x.DBX1.5) = 0 state. The timer for the controller enable delay time (MD36620 SERVO_DISABLE_DELAY_TIME (shutdown delay of controller enable)) is also started.
- As soon as the actual speed has reached the zero speed range, the drive controller enable is removed. Checkback signal to PLC with IS "Speed controller active" (DB390x.DBX1.6) = 0 state. The controller enable of the drive is removed at the latest after the time set in MD36620 SERVO_DISABLE_DELAY_TIME has expired.
- Notice: If the setting for the controller enable shutdown delay is too small the controller enable will be removed even though the axis/spindle is still moving. The axis/spindle is then stopped abruptly with setpoint 0.
- The actual position value of the axis/spindle continues to be acquired by the control.

This axis/spindle state cannot be changed until after "Reset".

Interpolatory axis grouping:

All the axes traversing within the interpolatory axis grouping are stopped as soon as the controller enable signal is cancelled for **one** of the axes.

The axes are brought to a standstill as described above. All axes in the geometry grouping are brought to a standstill with rapid stop. Alarm 21612 "Controller enable reset during movement" is also triggered. Continued processing of the NC program after this event is no longer possible.

2.2.3 Signals for digital drives, to axis/spindle

Speed controller integrator disabled (DB380x.DBX4001.6)

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

Pulse enable (DB380x.DBX4001.7)

The PLC user program enables the pulses for the axis/spindle. However, the pulse enable is only activated for the drive module if all the enable signals are present.

2.3 Signals from NCK to PLC

2.3.1 General signals

Drives in cyclic operation (DB2700.DBX2.5)

The PLC is signaled via the NCK by means of a cyclical exchange of data that the available drives have reached ramp-up status.

Drive ready (DB2700.DBX2.6)

The PLC is signaled via NCK that all available drives are ready to operate. IS "Drive Ready" (group signal) is active on all axes and spindles.

NCK alarm is active (DB2700.DBX3.0)

The control sends this signal to the PLC to indicate that at least one NCK alarm is active. An enquiry can be made via the channel-specific interface (DB3300.DBX4.7) as to whether a processing stop has been triggered.

Ambient temperature alarm (DB2700.DBX3.6)

The ambient temperature or fan monitoring function has responded.

NCK alarm channel-specific active (DB3300.DBX4.6)

The control system sends this signal to the PLC to indicate that at least one NCK alarm is active for the channel. To what extent this may influence whether the current program run will be interrupted or aborted can be determined from IS "NCK alarm with processing stop is active" (DB3300.DBX4.7).

External language mode active (DB3300.DBX4001.0)

The control system sends this signal to the PLC to indicate that the active program language used for the part program is not a SIEMENS language. A language changeover has been made with G291.

NCK alarm with processing stop present (DB3300.DBX4.7)

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

Follow-up active (DB390x.DBX1.3)

Follow-up mode for this axis is active.

See Section: Signals from PLC to NCK, follow-up mode (Page 15) (DB380x.DBX1.4)

Axis/spindle stationary (DB390x.DBX1.4)

The current velocity of the axis or actual speed of the spindle is within the range which is defined as standstill. This range is defined with MD36060 STANDSTILL_VELO_TOL (maximum velocity/speed for signal "Axis/spindle stationary").

Position control active (DB390x.DBX1.5)

The position control loop for the axis/spindle is closed; the position control function is active.

For details, see Controller enable (Page 15).

Speed control active (DB390x.DBX1.6)

The speed control loop for the axis/spindle is closed; the speed control function is active.

For details, see Controller enable (Page 15).

Current control active (DB390x.DBX1.7)

The current control loop for the axis/spindle is closed; the current control function is active.

Lubrication pulse (DB390x.DBX1002.0)

The IS "Lubrication pulse" is sent by the NCK and **changes status** once the axis/spindle has traveled a greater distance than that set in MD33050 LUBRICATION_DIST (travel distance for lubrication from PLC)

2.3.2 Signals for digital drives, from axis/spindle

Drive ready (DB390x.DBX4001.5)

Checkback signal indicating that the drive is ready. The conditions required for traversing the axis/spindle are fulfilled.

Integrator for n-controller disabled (DB390x.DBX4001.6)

The speed-controller integrator is disabled. The speed controller has thus been switched from PI to P controller.

Pulse enabled (DB390x.DBX4001.7)

The pulse enable for the drive module is available. The axis/spindle can now be traversed.

Ramp-up procedure completed (DB390x.DBX4002.2)

This signal confirms that the actual speed value has reached the new setpoint allowing for the tolerance band set in the drive. The ramp-up procedure is thus completed. Any subsequent speed fluctuations due to load changes will not affect the interface signal.

2.4 Signals from PLC to HMI

OP key disable (DB1900.DBX5000.2)

IS "OP key disable" can be applied to disable (1 signal) or enable (0 signal) the connected keyboard.

Program number (DB1700.DBB1000)

A declared program number is transferred from the PLC to HMI if an NC program is selected by the PLC. The current NC program selected can be stored via the command interface (see DB1700.DBB1001) and also selected again.

With SINUMERIK 808D, a program with the program name (STRING) is administered. In the assignment list, the names for a maximum of 255 programs can be declared and assigned.

The use of the numbers is divided into the protection areas of the programs:

- 1 to 100: User area (end user protection level)
- 101 to 200: Machine manufacturer (machine manufacturer protection level)
- 201 to 255: SIEMENS (SIEMENS protection level)

"Program number" (DB1700.DBB1000) corresponds to the following IS:

- "Program has been selected" (DB1700.DBX2000.0)
- "Program selection error" (DB1700.DBX2000.1).

When a program number > 0 is written, the program selection is started by the PLC. As soon as the HMI detects a program number > 0, it begins with the internal processing of this job and sets the program number (DB1700.DBB1000) to 0.

PLC waits until the acknowledgement signal from HMI is received: DB1700.DBX2000.0 or DB1700.DBX2000.1 and evaluates this immediately. The acknowledge signals are available for one PLC cycle once they have been received and are then automatically deleted by the PLC operating system.

Command (DB1700.DBB1001)

A command job is transferred from the PLC to the HMI.

Command	Action
0	None
1	Save name of the selected program
2	Select program with saved program name

"Command" (DB1700.DBB1001) corresponds to the following IS:

- "Execute command" (DB1700.DBX2001.0)
- "Command execution error" (DB1700.DBX2001.1)

When a command > 0 is written, the job is started by the PLC. As soon as the HMI detects a command > 0, it begins with the internal processing of this job and sets the command (DB1700.DBB1001) to 0.

PLC waits until the acknowledgement signal has been reached by HMI: DB1700.DBX2001.0 or DB1700.DBX2001.1 and evaluates this immediately. The acknowledgement signals are available for one PLC cycle once they have been received and are then automatically deleted by the PLC operating system.

2.5 Signals from HMI to PLC

Program has been selected (DB1700.DBX2000.0)

Successful selection of the required NC program is signaled back from the HMI to the PLC. This signal is available for one PLC cycle. It corresponds with DB1700.DBB1000.

Program selection error (DB1700.DBX2000.1)

Failed selection of the required NC program is signaled back from the HMI to the PLC. This signal is available for one PLC cycle. It corresponds with DB1700.DBB1000.

Execute command (DB1700.DBX2001.0)

Successful execution of the required command is signaled back from the HMI to the PLC. This signal is available for one PLC cycle. It corresponds with DB1700.DBB1001.

Command execution error (DB1700.DBX2001.1)

Failed execution of the required command is signaled back from the HMI to the PLC. This signal is available for one PLC cycle. It corresponds with DB1700.DBB1001.

2.6 User Interface

2.6.1 General (OF)

Communication jobs can be performed via the "NC services" PLC/NCK interface. The following services are available for this:

- Start program invocation services (PI services) in the NCK area (e.g. asynchronous subroutine (ASUP))
- Read variables from the NCK area
- Write variables from the NCK area

The activation of the respective service is performed via the global part of the interface. The parameterization of the individual services is described below.

Job, global part

Only one service can run at a time. The service is selected via DB1200.DBX0.1 and DB1200.DBX0.2:

Service	DB1200.DBX0.2	DB1200.DBX0.1
Start PI service in the NCK area	1	0
Read variables from the NCK area	0	0
Write variables from the NCK area	0	1

Start:

A job is started by setting the signal DB1200.DBX0.0 = 1. A new job can only be started if the previous job has been completed, i.e. the acknowledgement signals ("Job completed" DB1200.DBX2000.0 and "Error in job" DB1200.DBX2000.1) must be zero.

The execution of a job may take several PLC cycles and vary depending on the utilization; thus, this function is not real-time-capable.

Note

A job already started cannot be cancelled. If the "Start" signal is inadvertently reset before receiving the acknowledgement, the result signals for this job are not refreshed; the job, however, is executed.

Job, global part

The results are written by the PLC operating system; therefore, these signals can only be written by the user.

If the job was completed without errors, the "Job completed" signal DB1200.DBX2000.0 is set to 1. If an error occurs while executing a read/write job, the "error in job" signal DB1200.DBX2000.1 is set.

The result signals in DB1200.DBB2000 are global bits for the whole job. Possible error causes can be here, e.g.:

- Number of variables (DB1200.DBX1) outside of the valid range
- Variable index (DB1200.DBX1000) outside of the valid range

After evaluating the result, the "Start" signal (DB1200.DBX0.0) is reset by the user. The PLC operating system then resets "Job completed" or "Error in job".

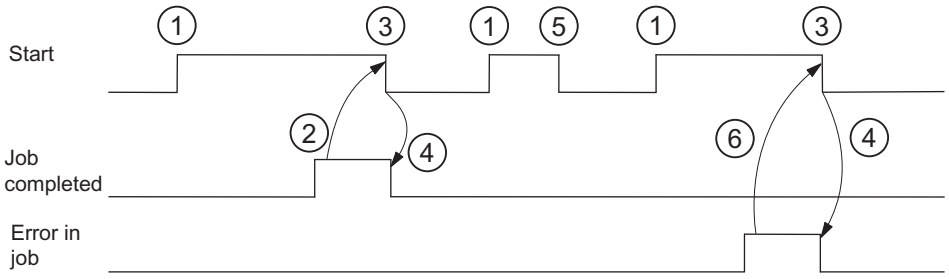


Figure 2-2 Pulse diagram

Explanations regarding the pulse diagram:

1. Starting of the job by setting "Start" ("Job completed" and "Error in job" must be reset)
2. Job completed without errors (the results of the individual variables must still be evaluated)
3. Resetting "Start" after receiving the result
4. Signal change by PLC operating system
5. If the "Start" signal is reset inadvertently before receiving the result, the output signals are not refreshed without influence on the internal execution of the function triggered
6. Error in job

2.6.2 PI service ASUP

Initialization

With the ASUP PI service, it is possible to assign the interrupt numbers 1 and 2 fixed program names from the PLC. Prerequisite for this is the existence of the PLCASUP1_SPF or PLCASUP2_SPF programs in the CMA directory.

PI index	Function
DB1200.DBB4001 = 1	Assignment of Interrupt 1 to the CMA_DIR/PLCASUP1_SPF program. The interrupt has Priority 1.
DB1200.DBB4001 = 2	Assignment of Interrupt 2 to the CMA_DIR/PLCASUP2_SPF program. The interrupt has Priority 2.

The following must be taken into account during the initialization:

- The PI service ASUP requires executing only once after a restart and is then retained.
- An initialization may only be performed when the channel is not active.
- If a "Ramp-up" program event has been configured, the initialization may only be started after the end of the program event.

Relevant interface signals

	Address	Name	Valid values
Job	DB1200.DBX4000.0	Start	0/1
	DB1200.DBX4000.1	Write variable	0
	DB1200.DBX4000.2	PI service	1
	DB1200.DBB4001	PI index	1,2
Result	DB1200.DBX5000.0	Request completed	0/1
	DB1200.DBX5000.1	Error in job	0/1

2.6.3 Reading variables from the NCK area

1 to 8 values can be read with a read job (variable x: 0...7). There is a variable-specific part of the interface for this:

- Job: DB120x.DBB1000
- Result: DB120x.DBB3000

Job, variable-specific part

NC variable:

The NC variable is selected in the variable index (DB120x.DBB1000), see Section: NC variable (Page 28)

Area number, column / line index (DB120x.DBB1001 ... DB120x.DBB1005)

Various variables are declared as fields. For flexible addressing, the relevant field index must be specified as a column and/or line index (e.g. R parameter no.).

Values:

The range DB120x.DBB1008 ... 11 is not relevant for reading.

Result, variable-specific part

A result is reported for each variable in the job.

If the read process was successful, "Variable valid" (DB120x.DBX3000.0) is set to 1; the access result DB120x.DBB3001 is 0.

When reading, the data from DB120x.DBB3004 are entered type-specifically.

In case of error, DB120x.DBX3000.0 remains "0", and an entry is made in the access result DB120x.DBB3001:

- 0: No error
- 3: Illegal access to object
- 5: Invalid address
- 10: Object does not exist

Values:

When reading, the read data are in the range DB120x.DBB3004...7, in the data type specific for the respective variable (if required, the values are converted from 64-bit to32-bit REAL).

Relevant interface signals

	Address	Name	Valid values
Job, global part	DB1200.DBX0.0	Start	0/1
	DB1200.DBX0.1	Write variable	0
	DB1200.DBX0.2	PI service	0
	DB1200.DBB1	Number of variables	1 ... 8

	Address	Name	Valid values
Job, variable-specific part	DB120x.DBB1000	Variable index	See Section NC variable (Page 28)
	DB120x.DBB1001	Area number	
	DB120x.DBB1002	Line index, NCK variable	
	DB120x.DBB1004	Column index, NCK variable	
Job, global part	DB1200.DBX2000.0	Request completed	0/1
	DB1200.DBX2000.1	Error in job	0/1
Result, variable-specific part	DB120x.DBX3000.0	Invalid variable	0/1
	DB120x.DBB3001	Access result	0/3/5/10
	DB120x.DBB3004/ DB120x.DBW3004/ DB120x.DBD3004	Value of NCK variable, data type depends on variable index	See Section NC variable (Page 28)

2.6.4 Writing variables from the NCK area

1 to 8 values can be written with a write job (variable x: 0...7). There is a variable-specific part of the interface for this:

- Job: DB120x.DBB1000
- Result: DB120x.DBB3000

Job, variable-specific part

NC variable:

The NC variable is selected in the variable index (DB120x.DBB1000), see Section: NC variable (Page 28)

Area number, column / line index (DB120x.DBB1001 ... DB120x.DBB1005)

Various variables are declared as fields. For flexible addressing, the relevant field index must be specified as a column and/or line index (e.g. R parameter no.).

Values:

The values to be written must be entered in the range DB120x.DBB1008...11 in the data type specific for the appropriate variable.

If necessary, the values are converted (e.g. NCL floating-point values (64-bit) into the PLC format (32-bit) and vice versa). A loss of accuracy results from the conversion from 64-bit to 32bit REAL. The maximum accuracy of 32bit REAL numbers is approximately 10^7.

Result, variable-specific part

A result is reported for each variable in the job.

If the read process was successful, "Variable valid" (DB120x.DBX3000.0) is set to 1; the access result DB120x.DBB3001 is 0.

When reading, the data as of DB120x.DBB3004 is entered type-specifically.

In case of error, DB120x.DBX3000.0 remains "0", and an entry is made in the access result DB120x.DBB3001:

- 0: No error
- 3: Illegal access to object
- 5: Invalid address
- 10: Object does not exist

Values:

The range DB120x.DBB3004...07 is not relevant for writing.

Relevant interface signals

	Address	Name	Valid values
Job, global part	DB1200.DBX0.0	Start	0/1
	DB1200.DBX0.1	Write variable	1
	DB1200.DBX0.2	PI service	0
	DB1200.DBB1	Number of variables	1 ... 8
Job, variable-specific part	DB120x.DBB1000	Variable index	See Section NC variable (Page 28)
	DB120x.DBB1001	Area number	
	DB120x.DBB1002	Line index, NCK variable	
	DB120x.DBB1004	Column index, NCK variable	
	DB120x.DBB3004/ DB120x.DBW3004/ DB120x.DBD3004	Value of NCK variable, data type depends on variable index	
Job, global part	DB1200.DBX2000.0	Request completed	0/1
	DB1200.DBX2000.1	Error in job	0/1
Result, variable-specific part	DB120x.DBX3000.0	Invalid variable	0/1
	DB120x.DBB3001	Access result	0/3/5/10

2.7 NC variable

Variable cuttEdgeParam

Compensation value parameters and cutting edge list with D numbers for a tool.

The meanings of the individual parameters depend on the type of the tool in question. Currently, totally 25 parameters are reserved for each tool edge (but only a part of them is loaded with values). To be able to remain flexible for future extensions, it is not recommended to use a fixed value of 25 parameters for calculation, but the variable value 'numCuttEdgeParams' (variable index 2).

For a detailed description of the tool parameters, please refer to Chapter "Tool Offset (Page 200)".

	Variable cuttEdgeParam [r/w]
DB120x.DBB1000	1
DB120x.DBB1001	-
DB120x.DBW1002	(EdgeNo - 1) * numCuttEdgeParams + ParameterNo (WORD)
DB120x.DBW1004	T number (1...32000) (WORD)
DB120x.DBD1008	Write: Data to NCK variable x (data type of the variables: REAL)
DB120x.DBD3004	Read: Data from NCK variable x (data type of the variables: REAL)

Variable numCuttEdgeParams

Number of P elements of an edge

	Variable numCuttEdgeParams [r]
DB120x.DBB1000	2
DB120x.DBB1001	-
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: WORD)

Variable linShift

Translation of a settable work offset (channel-specific settable frames)

They only exist if MD18601 MM_NUM_GLOBAL_USER_FRAMES > 0.

There are the frame indices:

- 0: ACTFRAME = current resulting work offset
- 1: IFRAME = current settable work offset
- 2: PFRAME = current programmable work offset
- 3: EXTFRAME = current external work offset

- 4: TOTFRAME = current total work offset = total of ACTFRAME and EXTFRAME
- 5: ACTBFRAME = current total base frame
- 6: SETFRAME = current 1st system frame (PRESET, scratching)
- 7: EXTSFRAME = current 2nd system frame (PRESET, scratching)
- 8: PARTFRAME = current 3rd system frame (TCARR and PAROT with orientable tool carrier)
- 9: TOOLFRAME = current 4th system frame (TOROT and TOFRAME)
- 10: MEASFRAME = result frame for workpiece and tool gauging
- 11: WPFRAME = current 5th system frame (workpiece reference points)
- 12: CYCFRAME = current 6th system frame (cycles)

The max. frame index is 12.

The value of numMachAxes is contained in the variable with variable index 4.

	Variable linShift [r]
DB120x.DBB1000	3
DB120x.DBB1001	-
DB120x.DBW1002	Frame index * numMachAxes + axis number
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBD3004	Read: Data from NCK variable x (data type of the variables: REAL)

Variable numMachAxes

No. of the highest existing channel axis

If there are no gap between channels, this corresponds to the number of existing axes in the channel.

	Variable numMachAxes [r]
DB120x.DBB1000	4
DB120x.DBB1001	-
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBW3004	Read: Data from NCK variable x (data type of the variables: WORD)

Variable rpa

R parameters

	Variable rpa [r/w]
DB120x.DBB1000	5
DB120x.DBB1001	-
DB120x.DBW1002	R number + 1
DB120x.DBW1004	-
DB120x.DBD1008	Write: Data to NCK variable x (data type of the variables: REAL)
DB120x.DBD3004	Read: Data from NCK variable x (data type of the variables: REAL)

Variable actLineNumber

Line number of the current NC block:

- 0: Prior to program start
- -1: Not available due to error
- -2: Not available due to `DISPLOF`

	Variable actLineNumber [r]
DB120x.DBB1000	6
DB120x.DBB1001	-
DB120x.DBW1002	-
DB120x.DBW1004	-
DB120x.DBD1008	-
DB120x.DBD3004	Read: Data from NCK variable x (data type of the variables: DINT)

2.8 Signals from PLC

Commissioning mode

The ramp-up modes are signaled via bit 0 and bit 1 (DB1800.DBB1000) in the user interface.

Commissioning mode	DB1800.DBX1000.1	DB1800.DBX1000.0
Normal rampup	0	0
Ramp-up with default values	0	1
Ramp-up with saved data	1	0

Axis Monitoring

3

3.1 Overview of monitoring functions

Overview of monitoring functions

- Motion monitoring functions
  - Contour monitoring
  - Position monitoring
  - Standstill monitoring
  - Clamping monitoring
  - Speed setpoint monitoring
  - Actual velocity monitoring
  - Encoder monitoring functions
- Monitoring of static limits
  - Limit switch monitoring

3.2 Running monitoring

3.2.1 Contour monitoring

Function

The principle on which the contour monitoring function works is the constant comparison of the measured actual position value with that calculated from the NC position setpoint. For the precalculation of the following error, a model is used that simulates the dynamics of the position control including feedforward control.

So that the monitoring function does not respond incorrectly on slight speed fluctuations (caused by changes of load) a tolerance band is allowed for the max. contour deviation.

If the permissible actual value deviation entered in MD36400 CONTOUR_TOL (tolerance band contour monitoring) is exceeded, an alarm is signaled and the axes are stopped.

Effectiveness

Contour monitoring is active for axes and position-controlled spindles.

Effect

If the contour deviation is too large, this has the following effect:

- Alarm 25050 "Contour monitoring" is triggered
- The axis/spindle is brought to a standstill via a speed setpoint ramp with rapid stop (with open position control loop).
  The braking ramp time is set in
  MD36610 AX_EMERGENCY_STOP_TIME (braking ramp time for error states).
- If the axis/spindle is involved in interpolation with other axes/spindles, these are brought to a standstill with rapid stop with following error reduction (position setpoint = 0).

Remedy

- Increase tolerance band of monitoring in MD36400
- The actual "servo gain factor" must correspond to the desired servo gain factor set via MD32200 POSCTRL_GAIN (servo gain factor).
  With analog spindles:
  MD32260 RATED_VELO (rated motor speed) and
  MD32250 RATED_OUTVAL (rated output voltage) must be checked.
- Check optimization of the speed controller
- Check smooth running of the axes
- Check machine data for traversing movements
  (feed override, acceleration, max. speeds, ...)

3.2.2 Position monitoring

Function

In order to ensure that an axis reaches the required position within the specified time, the timer that can be configured in MD36020 POSITIONING_TIME (time delay exact stop fine) is started at the end of each motion block (setpoint has reached target) and, when the timer runs out, a check made to ascertain whether the axis has reached its setpoint within the tolerance of MD36010 STOP_LIMIT_FINE (exact stop fine).

For details on "Exact stop coarse and fine" see Chapter "Continuous Path Mode, Exact Stop and Look-Ahead (Page 45)"

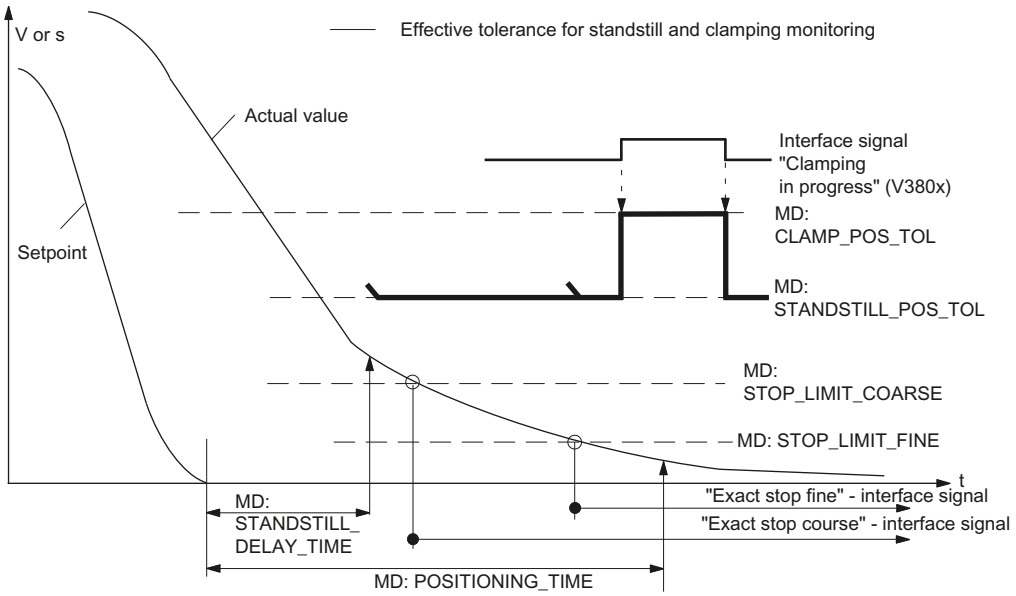


Figure 3-1 Relation between position, standstill, and clamping monitoring

Effectiveness

Positioning monitoring is always activated after the termination of motion blocks "according to the setpoint" (setpoint has reached destination).

Position monitoring is active for axes and position-controlled spindles.

Deactivation

When the programmed "Exact stop limit fine" has been reached or a new setpoint has been output (e.g. for positioning according to "Exact stop coarse" followed by a block change), the position monitoring is deactivated.

Effect

If the limit value for "Exact stop fine" has not yet been reached when the positioning monitoring time has elapsed, the following action is performed:

- Output of alarm 25080 "Positioning monitoring"
- The affected axis/spindle is brought to a standstill using a rapid stop (with open position control loop) along a speed setpoint ramp.
  The braking ramp duration is set in
  MD36610 AX_EMERGENCY_STOP_TIME (braking ramp duration for error states).
- If the axis/spindle is involved in interpolation with other axes/spindles, these are stopped using a rapid stop with following error reduction (default for partial position setpoint = 0).

Cause of error/Remedy

- Position controller gain too low --> change machine data for position controller gain
  MD32200 POSCTRL_GAIN (servo gain factor)
- Positioning window (exact stop fine), position monitoring time, and position controller gain have not been coordinated --> change machine data:
  MD36010 STOP_LIMIT_FINE (exact stop fine),
  MD36020 POSITIONING_TIME (exact stop fine delay time),
  MD32200 POSCTRL_GAIN (servo gain factor)

Rule of thumb

- Positioning window large --> max. position monitoring time can be set to a relatively short value
- Positioning window small --> max. position monitoring time must be set to a relatively long value
- Position controller gain low --> max. position monitoring time must be set to a relatively long value
- Position controller gain high --> max. position monitoring time can be set to a relatively short value

Note

The size of the positioning window affects the block change time. The smaller the tolerances that are selected, the longer the positioning action will take, which in turn means a longer time before the next command can be executed.

3.2.3 Standstill monitoring

Function

At the end of a motion block (position setpoint has reached target), a check is made as to whether the axis is not more than the distance specified in MD36060 STANDSTILL_POS_TOL (standstill tolerance) away from its setpoint after the configurable delay time in MD36040 STANDSTILL_DELAY_TIME (standstill monitoring delay time) has expired. Otherwise, an alarm will be triggered.

Effectiveness

Standstill monitoring is always active after "Standstill monitoring delay time" active has expired, as long as no new travel command is present.

Standstill monitoring is active on axes and position-controlled spindles.

Effect

When the monitoring function responds, it has the following effects:

- Alarm 25040 "Standstill monitoring" is triggered
- The affected axis/spindle is brought to a standstill with rapid stop (with open position control loop) along a speed setpoint ramp. The braking ramp time is set in MD36610 AX_EMERGENCY_STOP_TIME (duration of the braking ramp for error states).
- If the axis/spindle is involved in interpolation with other axes/spindles, these are stopped by rapid stop with following error reduction (default for position partial setpoint = 0).

Cause of error / remedy

- Position control gain too high (control loop oscillation) --> change machine data for control gain MD32200 POSCTRL_GAIN (servo gain factor)
- Standstill window too small --> change machine data MD36030 STANDSTILL_POS_TOL (standstill tolerance)
- Axis is mechanically "pushed" out of position --> eliminate cause

3.2.4 Clamping monitoring

Function

If the axis must be clamped once it has been positioned, the clamping monitoring function can be activated via IS "Clamping in progress" (DB380x.DBX2.3).

This may be necessary as the axis can be forced further from the setpoint than the standstill tolerance permits during the clamping process. The amount by which the axis may leave the command position is specified in MD36050 CLAMP_POS_TOL (clamping tolerance for interface signal "Clamping active").

Effectiveness

Clamping monitoring is activated by the interface signal "Clamping active". It replaces standstill monitoring during clamping.

Clamping monitoring is active on axes and position-controlled spindles.

Effect

If the axis is pushed out of position beyond the clamping tolerance during clamping the following occurs:

- Alarm 26000 "Clamping monitoring" is triggered
- The affected axis/spindle is brought to a standstill with rapid stop (with open position control loop) along a speed setpoint ramp. The braking ramp time is set in MD36610 AX_EMERGENCY_STOP_TIME (duration of the braking ramp for error states).
- If the axis/spindle is assigned to an interpolatory grouping with other axes/spindles, then these are also braked by rapid stop with following error reduction (default for partial position setpoint = 0).

3.2.5 Speed setpoint monitoring

Function

Speed setpoint monitoring checks whether the setpoint specification does not exceed the maximum permissible drive speed in MD 36210 CTRLOUT_LIMIT (maximum speed setpoint). If required, the speed is limited to this value and the axis/spindle stopped and an alarm output.

The maximum speed for the axes (in percent) exceeds the speed at which the velocity in MD32000 MAX_AX_VELO is reached (100%). This also determines the control margin.

On an analog spindle the maximum speed that can be output must not exceed the speed reached at the maximum setpoint output voltage of 10 V (100%).

The speed setpoint consists of the speed setpoint of the position controller and the feedforward control parameter (if feedforward control is active).

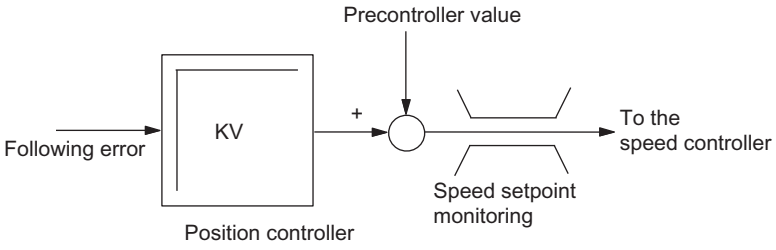


Figure 3-2 Speed setpoint calculation

Effectiveness

Speed setpoint monitoring is always active for axes and spindles.

Effect

The following occurs if the maximum speed setpoint value is exceeded:

- Alarm 25060 "Speed setpoint limiting" is triggered
- The affected axis/spindle is brought to a standstill using a rapid stop (with open position control loop) along a speed setpoint ramp.

  The braking ramp duration is set in MD36610 AX_EMERGENCY_STOP_TIME (braking ramp duration for error states).
- If the axis/spindle is involved in interpolation with other axes/spindles, these are stopped using a rapid stop with following error reduction (default for partial position setpoint = 0).

Note

In the "Expert mode" access level (protection level 1), MD36220 CTRLOUT_LIMIT_TIME can be used to set a delay time, after the expiration of which an alarm is output and the axes are brought to a standstill. The default value of this time is zero.

Using speed setpoint limiting will turn the control loop into a non-linear control loop. This generally causes contour deviations if speed setpoint limiting is continued for an axis. A control margin must therefore be set.

Causes of errors

- A measuring circuit error or drive error is present.
- Setpoints are too high (accelerations, velocities, reducing factors).
- Obstacle in work area (e.g. positioning on a working table)
- Tachogenerator compensation has not been performed correctly for an analog spindle, or a measuring circuit error or drive error is present.

3.2.6 Actual velocity monitoring

Function

This function monitors whether the actual velocity exceeds a permissible limit entered in MD36200 AX_VELO_LIMIT (threshold value for velocity monitoring).

Effectiveness

The actual velocity monitor is operative whenever the active measuring circuit activated via "Position measuring system 1" interface signal (DB380x.DBX1.5) is supplying actual values, i.e. still operating below the limit frequency.

The actual velocity monitoring is active for axes and spindles.

Effect

If the "Threshold for velocity monitoring" is exceeded the following occurs:

- Alarm 25030 "Actual velocity alarm limit" is triggered
- The affected axis/spindle is brought to a standstill with rapid stop (with open position control loop) along a speed setpoint ramp. The braking ramp time is set in MD36610 AX_EMERGENCY_STOP_TIME (duration of the braking ramp for error states).
- If the axis/spindle is assigned to an interpolatory grouping with other axes/spindles, then these are also braked by rapid stop with following error reduction (default for partial position setpoint = 0).

Troubleshooting tips

- Check actual values
- Check position control direction (control sense)
- Check MD36200 AX_VELO_LIMIT (threshold value for velocity monitoring)
- Check signal setpoint cable for analog spindles

3.3 Static limitation monitoring

3.3.1 Limit switch monitoring

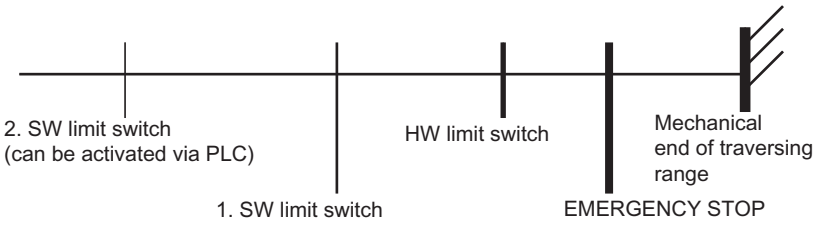


Figure 3-3 Overview of travel limits of a linear axis

3.3.2 Hardware limit switches

Function

Every axis has a hardware (HW) limit switch for each traversing direction, which prevents the slide from moving out of the slide bed.

If the hardware limit switch is crossed, the PLC signals this to the NC via IS "Hardware limit switch plus/minus" (DB380x.DBX1000.1 or .0) and the movement of all axes is stopped. The braking method can be specified via MD36600 BRAKE_MODE_CHOICE (braking behavior at hardware limit switch).

Effectiveness

HW limit switch monitoring is active after the control has started up in all modes.

Effect

- When a hardware limit switch is passed in either direction, alarm 21614 "Hardware limit switch + or -" is triggered.
- The axis is stopped according to the setting in MD36600 BRAKE_MODE_CHOICE (braking behavior at hardware limit switch).
- If the axis is assigned to an interpolatory grouping with other axes, then these are also stopped according to the method selected in MD36600 BRAKE_MODE_CHOICE (braking behavior at hardware limit switch).
- The direction keys in the approach direction are disabled.

Remedy

- Reset
- Move in the opposite direction (in JOG mode)
- Correct the program

3.3.3 Software limit switches

Function

They are used to limit the maximum traversing range on each individual axis.

There are two pairs of software limit switches for each machine axis. They are defined in the machine axis system using the following machine data:

MD36100 POS_LIMIT_MINUS (1st software limit switch minus)

MD36110 POS_LIMIT_PLUS (1st software limit switch plus)

MD36120 POS_LIMIT_MINUS2 (2nd software limit switch minus)

MD36130 POS_LIMIT_PLUS2 (2nd software limit switch plus)

Effectiveness

- Software (SW) limit switch monitoring is activated after reference point approach in all modes.
- The position of the software limit switch can be approached.
- The 2nd software limit switch can be activated via the "2nd software limit switch plus/minus" interface signal (DB380x.DBX1000.3 or .2) from the PLC. The change becomes active immediately. The 1st software limit switch plus/minus is then de-activated.
- The SW limit switch monitoring does not function for endlessly turning rotary axes, i.e. if MD30310 ROT_IS_MODULO = 1. (Modulo conversion for rotary axis and spindle)

Effect/reactions

Based on the mode, different responses to an attempted software limit switch violation are possible:

AUTO, MDA:

- The block that would violate the software limits switches is not started. The previous block is terminated properly.
- Program execution is terminated.
- Alarm 10720 "Software limit switch + or -" is signaled.

JOG:

- The axis stops at the software limit switch position.
- Alarm 10621 "Axis at software limit switch + or -" is signaled.
- The direction keys in the approach direction are disabled.

Note

Switching over the software limit switch:

If the current position lies behind the new software limit switch when the software limit switch is switched over, the axis is decelerated with the maximum permissible axial acceleration. If an axis is involved in interpolation with other axes, these are also decelerated. Then a contour violation may occur.

Remedy

- Reset
- Move in the opposite direction (in JOG mode)
- Correct the program

3.4 Supplementary conditions

To ensure that the monitoring functions respond correctly, it is important that the correct values are entered in the following machine data:

General:

- MD31030 LEADSCREW_PITCH (leadscrew pitch)
- Gear ratio (load gearbox):

  MD31050 DRIVE_AX_RATIO_DENOM (load gearbox denominator)

  MD31060 DRIVE_AX_RATIO_NUMERA (load gearbox numerator)

  Gear ratio (encoder), possibly for spindle:

  MD31070 DRIVE_ENC_RATIO_DENOM (measuring gearbox denominator)

  MD31080 DRIVE_ENC_RATIO_NUMERA (measuring gearbox numerator)
- MD32810 EQUIV_SPEEDCTRL_TIME

  (Equivalent time constant speed control loop for feedforward control)
- Encoder resolution

  MD31020 ENC_RESOL[0] (encoder pulses per revolution)

The associated machine data are described in Chapter "Velocities, Setpoint/Actual Value Systems, Closed-Loop Control (G2)"

For analog spindle only:

- Output voltage / output speed relation

  MD32260 RATED_VELO (rated motor speed)

  MD32250 RATED_OUTVAL (rated output voltage)

3.5 Data table

3.5.1 Machine data

Number	Identifier	Name
Axis/spindle-specific		
30310	ROT_IS_MODULO	Modulo conversion for rotary axis and spindle
32000	MAX_AX_VELO	Maximum axis velocity
32200	POSCTRL_GAIN [n]	Servo gain factor Kv
32250	RATED_OUTVAL	Rated output voltage
32260	RATED_VELO	Rated motor speed
32300	MAX_AX_ACCEL	Axis acceleration
32810	EQUIV_SPEEDCTRL_TIME [n]	Equivalent time constant speed control loop for feedforward control
36000	STOP_LIMIT_COARSE	Exact stop coarse
36010	STOP_LIMIT_FINE	Exact stop fine
36020	POSITIONING_TIME	Time delay exact stop fine
36030	STANDSTILL_POS_TOL	Standstill tolerance
36040	STANDSTILL_DELAY_TIME	Delay time standstill monitoring
36050	CLAMP_POS_TOL	Clamping tolerance with "Clamping active" interface signal
36060	STANDSTILL_VELO_TOL	Maximum velocity/speed "Axis/spindle stationary"
36100	POS_LIMIT_MINUS	1. Minus software limit switch
36110	POS_LIMIT_PLUS	1. Plus software limit switch
36120	POS_LIMIT_MINUS2	2. Minus software limit switch
36130	POS_LIMIT_PLUS2	2. Plus software limit switch
36200	AX_VELO_LIMIT [n]	Threshold value for velocity monitoring
36210	CTRLOUT_LIMIT[n]	Maximum speed setpoint
36300	ENC_FREQ_LIMIT n	Encoder frequency limit
36302	ENC_FREQ_LIMIT_LOW	Encoder limit frequency resynchronization
36310	ENC_ZERO_MONITORING [n]	Zero mark monitoring
36400	CONTOUR_TOL	Tolerance band contour monitoring
36500	ENC_CHANGE_TOL	High backlash values / Maximum tolerance for actual position value changeover
36600	BRAKE_MODE_CHOICE	Braking behavior at hardware limit switch
36610	AX_EMERGENCY_STOP_TIME	Length of the braking ramp for error states
36620	SERVO_DISABLE_DELAY_TIME	Cutout delay controller enable

3.5.2 Interface signals

Number	.Bit	Name
Axis/spindle-specific		
DB380x.DBX1	.5	Position measuring system 1
DB380x.DBX2	.3	Clamping in progress
DB380x.DBX1000	.0 / .1	Hardware limit switch minus / hardware limit switch plus
DB380x.DBX1000	.2 / .3	2. Software limit switch minus / software limit switch plus
DB390x.DBX0	.2	Encoder limit frequency exceeded 1
DB390x.DBX0	.4	Referenced/synchronized 1

Continuous Path Mode, Exact Stop, and LookAhead 4

4.1 Brief description

For continuous path control, the Computerized Numerical Control (CNC) processes a part program block by block. Only when the functions of the current block have been completed is the next block processed. Various requirements with respect to machining or positioning require different block change criteria. There are two ways that the path axes can behave at block boundaries.

The first way is called "exact stop" and means that all path axes must have reached the set target position depending on an exact-stop criterion before the next block change is initiated. To be able to fulfill the criterion, the path axes must reduce the path velocity at every block change which, however, delays the block change.

The second way is called "continuous path mode" and it attempts to avoid deceleration of the path velocity at the block boundary in order to change to the next block with as little change of path velocity as possible.

"LookAhead" is a procedure in continuous path mode that achieves velocity control with LookAhead over several NC part program blocks.

4.2 General

Machine axes that are related interpolatively must have the same dynamic response, i.e. the same following error at any given velocity.

The term path axes refer to all machining axes which are controlled by the interpolator calculating the path points in such a manner that:

- All the axes involved start at the same time
- All the axes involved travel with the correct velocity ratios
- All the axes reach the programmed target position at the same time

The acceleration rates of the individual axes may vary depending on the path, e.g. circular path.

Path axes can be geometry axes and special axes (e.g. workpiece turning axes that are involved in the workpiece machining process).

Velocity for zero cycle blocks

The term zero cycle is applied to blocks whose path length is shorter than the distance that can be traveled on the basis of the programmed set feedrate and the interpolator cycle (time). For reasons of precision the velocity is reduced until at least one interpolator cycle is required for the distance. The velocity is then equal to or less than the quotient of the path length of the block and the interpolator (IPO) cycle.

Stop for synchronization

Regardless of whether exact stop or continuous path mode is selected, the block change can be delayed by synchronization processes which can stop the path axes. In exact stop mode, the path axes are stopped at the end of the current block. In continuous path mode, the path axes are stopped at the next block end point at which they can be decelerated without violating their deceleration limits. The following synchronization processes cause axes to stop.

- PLC acknowledgment

  If acknowledgment by the PLC is required for an auxiliary function that is output before or after the end of motion, the axes stop at the end of the block.

- Missing following blocks

  If following blocks are conditioned too slowly (e.g. "External processing") the axes stop at that last possible block boundary.

- Emptying of the buffer

  If the NC part program requests that the run-in be synchronized with the main run (empty the buffer, e.g. STOPRE), this involves an implicit block-related velocity reduction or exact stop.

Stopping because of synchronization does not cause contour violations. However, stopping is undesirable, especially in continuous path mode because it can cause backing off.

4.3 Exact stop

With the exact stop function (G60, G9), all the path axes must reach the programmed block end point. Only when all path axes have reached the exact stop criterion is the block change performed. The velocity at the block transition is practically zero.

That is:

- The path axes at the block end point are decelerated almost to rest without overshoot.
- The delay for fulfilling the exact stop criterion prolongs the machining time.
- The delay for fulfilling the exact stop criterion can cause backing off.

The use of the exact stop function is suitable for precise traversing of contours.

Exact stop is not suitable if

- Exact traversing of the contour on the basis of the criterion (e.g. exact stop fine) can deviate from the programmed contour in order to achieve faster machining.
- An absolutely constant velocity is required.

Activate exact stop

The "Exact stop" function can be selected in the NC part program by command G60 or G9. G60 is modal, G9 is non-modal. G9 is used if continuous path mode is to be interrupted. Both exact stop functions only function with the selected exact stop criterion (G601, G602). The "exact stop" function is de-selected with the continuous path mode function (G64).

Exact-stop criteria

- Exact stop fine: G601

  This criterion is applied to monitor whether the actual/setpoint position deviation of the axis has remained within a specific distance. The value of the permissible distance is stored in MD36010 STOP_LIMIT_FINE (exact stop fine)

- Exact stop coarse: G602

  Functions as exact stop fine, although the monitoring window is stored in MD36000 STOP_LIMIT_COARSE (exact stop coarse). To permit a faster block change than with the exact stop fine criterion, the exact stop coarse criterion is set to be larger than the exact stop fine criterion.

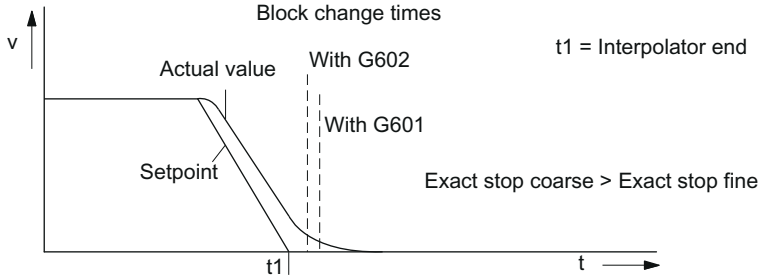


Figure 4-1 Block change depending on exact-stop criteria

Interpolator end

Interpolator end is achieved when the interpolator has calculated the setpoint velocity of the axes from zero for an interpolation cycle. However, the actual positions of the path axes have not reached the target (following error).

Irrespective of continuous-path mode or the active exact-stop criteria for the exact-stop function, "interpolator end" transfers the auxiliary functions present in the block to the PLC if they are to be output after the end of motion.

4.4 Continuous path mode

4.4.1 General

In continuous path mode, the path velocity is not decelerated for the block change in order to permit the fulfillment of an exact stop criterion. The objective of this mode is to avoid rapid deceleration of the path axes at the block-change point so that the axis velocity remains as constant as possible when the program moves to the next block. To achieve this objective, the "LookAhead" function is also activated when continuous path mode (G64) is selected.

Continuous path mode causes:

- Rounding of the contour.
- Shorter machining times through elimination of braking and acceleration processes that are required to comply with the exact-stop criterion.
- Improved cutting conditions as the velocity is more uniform.

The continuous-path mode is suitable if a contour is to be traversed as quickly as possible.

Continuous-path mode is suitable if:

- A contour is to be traversed precisely.
- An absolutely constant velocity is required.

Implicit exact stop

In some cases, an exact stop needs to be generated in continuous path mode to allow the execution of subsequent actions. In such situations, the path velocity is reduced to zero.

- If auxiliary functions are output before the traverse motion, the previous block is only terminated when the selected exact-stop criterion is fulfilled.
- If auxiliary functions are to be output after the traverse motion, they are output after the interpolator end of the block.
- If an executable block contains no travel information for the path axes, the previous block is terminated on reaching the selected exact stop criterion.
- A block is terminated on interpolator end if the following block contains the changeover of the acceleration profile BRISK/SOFT.
- If the function "Empty buffer" (STOPRE) is programmed, the previous block is terminated when the selected exact stop criterion is reached.

Velocity = 0 in continuous path mode

Regardless of the implicit exact stop response, the path motion is braked down to zero velocity at the end of the block in cases where:

- The time taken to position a spindle programmed with SPOS is longer than the travel time of the path axes. The block change is carried out when the "exact stop fine" of the positioning spindle is reached.
- A synchronization process needs to be carried out (see Section "General (Page 45)").

Auxiliary function output during traversal

If the traversal time is not sufficient due to the programmed path length and velocity of the block with auxiliary function output, the path velocity for the block is reduced such that the acknowledgment of the auxiliary function can arrive with a PLC cycle time.

If the acknowledgment is not received within one PLC cycle time, the following prepared block cannot be processed and the axes are braked to rest with setpoint = 0 (without considering the acceleration limits).

If the acknowledgment is not received by the end of the block in long blocks in which the velocity has not needed to be reduced on account of the PLC acknowledgment time, the velocity is maintained until the end of the block and then reduced as described above.

If the acknowledgment arrives while the axis is decelerating, the axis is not accelerated back up to the requested velocity.

4.4.2 Velocity reduction according to overload factor

Function

This function lowers the path velocity in continuous path mode until the nontangential **block transition** can be traversed in one interpolation cycle whilst respecting the deceleration limit and taking an overload factor into account. With the reduced velocity, axis-specific jumps in velocity are produced with a nontangential contour at the block transition. The jump in velocity prevents the path velocity dropping to zero. This jump is performed if the axial velocity was reduced with the axial acceleration to a velocity from which the new setpoint can be reached with the jump.

The magnitude of the setpoint jump can be limited using an overload factor. Because the magnitude of the jump is axial, the minimum jump of the path axes which are active during the block change is considered during block transition. With a practically tangential block transition, the path velocity is not reduced if the permissible axial accelerations are not exceeded. In this way, very small angular changes in the contour can be overtraveled directly.

Overload factor

The overload factor restricts step changes in the machine axis velocity at the block transition. To ensure that the velocity jump does not exceed the maximum load on the axis, the jump is derived from the acceleration of the axis. The overload factor indicates the extent by which the acceleration of the machine axis, which is set in MD32300 MAX_AX_ACCEL (axis acceleration), may be exceeded for an IPO cycle.

The velocity jump is the product of:

axis acceleration * (overload factor-1) * interpolator cycle. The overload factor is 1.2.

Factor 1.0 means that only tangential transitions with finite velocity can be traversed. For all other transitions, the velocity is reduced to zero by changing the setpoint.

Selection and deselection of velocity reduction

Continuous-path mode with velocity reduction according to overload factor can be selected modally in every NC part program block by means of program code G64 (BRISK active, not SOFT).

Continuous path mode G64 can be

- interrupted non-modally when exact stop G9 is selected,
- de-selected when exact stop G60 is selected.

4.4.3 Jerk limiting along the path through velocity reduction

Introduction

With the jerk limiting along the path, another method of influencing the continuous-path mode is introduced. While the "Velocity reduction according to overload factor" function limits the rate of velocity change, the "Jerk limitation on path" function described here limits the acceleration changes (jerks).

When sections of the contour consisting of blocks (e.g. circle straight line transitions) are machined, step changes in the acceleration rate occur at the **block transition** in continuous path mode.

Reducing jerk

The severity of such jerks can be reduced by decreasing the path velocity at transitions between blocks containing different degrees of curvature. A smoother transition is thus achieved between the contour sections.

Jerk limit

The user specifies the maximum jerk, which may occur on a path axis during a block transition, with MD32432 PATH_TRANS_JERK_LIM (maximum axis-specific jerk of a path axis at the block transition).

Activating

Jerk limiting at block transitions becomes active if continuous path mode is programmed with G64 and SOFT acceleration characteristics. MD32432 PATH_TRANS_JERK_LIM must contain a positive value.

4.4.4 Machine axis-specific jerk limiting

Function

The axis-specific machine data MD32431 MAX_AX_JERK[..] can be used to set individual changes in acceleration for each machine axis, like those that can already be set for acceleration limits in machine data MD32300 MAX_AX_ACCEL.

MD32431 MAX_AX_JERK acts on the axes interpolated by the path if **SOFT** (smooth acceleration curve) is active **within a block**.

A basic distinction is made between the axis acceleration curve within a block and at the transition between two blocks.

Advantages

The deployment of axis-specific machine data for the path offers the following advantages:

- Immediate allowance is made in the interpolation for the dynamic response of the axes, which can then be fully utilized for each axis.
- Jerk limitation for separate axes is performed not just in linear blocks, but also in curved contours.

Please refer to Chapter "Acceleration (Page 55)" for more information on the subject of "jerk limiting".

4.5 LookAhead

Function

LookAhead is a procedure in continuous path mode (G64) that achieves velocity control with LookAhead over several NC part program blocks beyond the current block.

Without LookAhead: If the program blocks only contain very small paths, a velocity per block is achieved that permits deceleration of the axes at the block end point without violating acceleration limits. This means that the programmed velocity was not actually reached although a sufficient number of prepared blocks with virtually tangential path transitions were available.

With the LookAhead function: It is possible to plan the acceleration and deceleration phase with approximately tangential path transitions over several blocks in order to achieve a higher feedrate with shorter distances. Deceleration to velocity limits is possible with LookAhead such that violation of the acceleration and velocity limit is prevented.

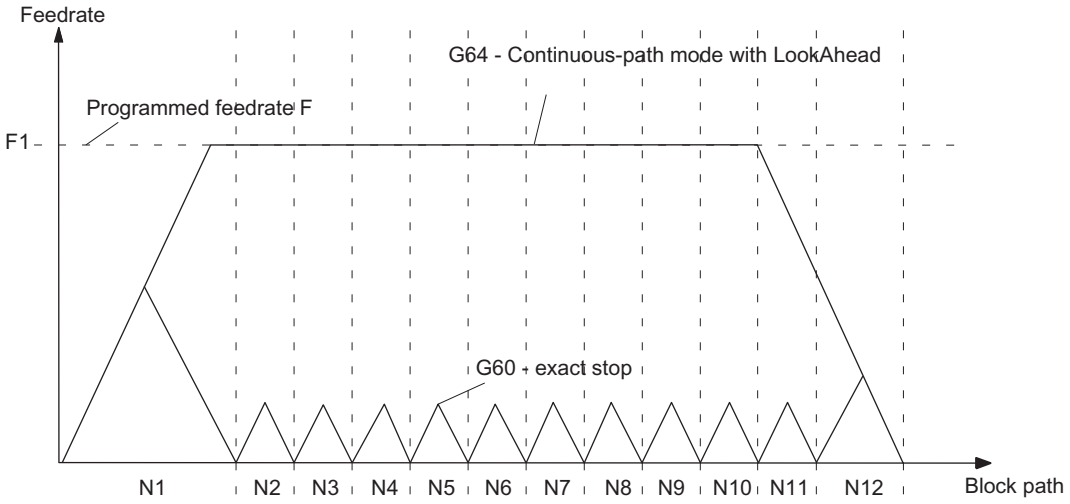


Figure 4-2 Comparison of the G60 and G64 velocity behavior with short travels in the blocks

LookAhead takes plannable velocity limits into consideration such as:

- Velocity limit in the block
- Acceleration limit in the block
- Velocity limit on block transition
- Synchronization with block change at block transition
- Exact stop at block end during termination

Operating principle

LookAhead functionality is available only for path axes, but not for the spindle.

For safety reasons, the velocity at the end of the last prepared block must initially be assumed to be zero because the next block might be very small or be an exact-stop block and the axes must have been stopped by the end of the block. With a series of blocks with high set velocity and very short paths, the speed can be increased in each block depending on the velocity value currently calculated by the LookAhead function in order to achieve the required set velocity. After this it can be reduced so that the velocity at the end of the last block considered by the LookAhead function can be zero. This results in a sawtooth-shaped velocity profile which can be avoided by reducing the set velocity for the number of blocks considered by the LookAhead function (fixed value).

Velocity profiles

In addition to the fixed, plannable velocity limitations, LookAhead can also take account of the programmed velocity. This makes it possible to achieve a lower velocity by applying LookAhead beyond the current block.

Following block velocity

One possible velocity profile contains the determination of the following block velocity. Using information from the current and the following NC block, a velocity profile is calculated from which, in turn, the required velocity reduction for the current override is derived. The calculated maximum value of the velocity profile is limited by the maximum path velocity.

With this function it is possible to initiate a speed reduction in the current block taking override into account such that the lower velocity of the following block can be achieved. If the reduction in velocity takes longer than the travel time of the current block, the velocity is further reduced in the following block. Velocity control is only ever considered for the following block.

Selection and deselection of LookAhead

If the continuous-path mode (G64) is selected LookAhead is called and de-selected/interrupted with G60/G9.

4.6 Data table

4.6.1 Machine data

Number	Identifier	Name
Channel-specific		
29000	LOOKAH_NUM_CHECKED_BLOCKS	Number of blocks considered by the LookAhead function
Axis/spindle-specific		
32431	MAX_AX_JERK	Maximum axis-specific jerk for path movement
32432	PATH_TRANS_JERK_LIM	Maximum axis-specific jerk for path movement at block transition
36000	STOP_LIMIT_COARSE	Exact stop coarse
36010	STOP_LIMIT_FINE	Exact stop fine
36020	POSITIONING_TIME	Delay time exact stop fine

4.6.2 Interface signals

Number	Bit	Name
Channel-specific		
DB3300.DBX0004	.3	All axes stationary
Axis/spindle-specific		
DB390x.DBX0000	.6	Position reached with exact stop coarse
DB390x.DBX0000	.7	Position reached with exact stop fine

Acceleration 5

5.1 Acceleration profiles

Abrupt acceleration changes

With the v/t-linear control of the axis velocity that is normally applied, the motion is controlled such that the acceleration rate changes abruptly over time. With the discontinuous, stepped acceleration, jerk-free starting and braking of the axes is not possible, but a time optimized velocity/time profile can be implemented.

Acceleration with jerk limitation

The jerk is the change of acceleration over time. For jerk-limited acceleration the maximum acceleration is not abrupt, but is specified by a ramp. Because of the softer acceleration progression, the traverse time is longer than with abrupt acceleration for the same distance, velocity and acceleration. This time loss can be compensated for by setting a higher acceleration for the axes.

However, it has the following advantages:

- Reduced wear to mechanical parts of the machine
- Reduction of the excitation of high frequency, difficult to control vibrations of the machine.

5.2 Jerk limitation on interpolator level

Selection and deselection of jerk-limited acceleration

MD32431 MAX_AX_JERK (maximum axis-specific jerk during path motion) can limit the change in acceleration per machine axis individually. It only acts on the axes interpolated by the path when SOFT is active. Jerk limitation is implemented entirely on the interpolator level.

Acceleration with jerk limitation is activated by:

The program code **SOFT** in the part program. SOFT is modal and causes deselection of the abrupt acceleration profile (BRISK). If SOFT is programmed in a block with path axes, the previous block is ended with exact stop.

Acceleration with jerk limitation (SOFT) is deactivated by:

The program code **BRISK** in the part program. BRISK is modal. If path axes are programmed in a block with BRISK, the previous block is ended with exact stop. BRISK activates the profile with abrupt acceleration changes associated with v/t-linear velocity control.

Applicability

Path-related jerk limitation is available for interpolating path axes in operating modes "AUTO" and "MDA". The SOFT and BRISK acceleration profiles can be used in traverse modes exact stop G9, G60, continuous path modes G64, and with LookAhead. The profiles are also active with the dry run feedrate function. With alarms that trigger a rapid stop, both acceleration profiles are inactive.

Further information about velocity, acceleration and jerk whilst traversing in continuous path mode and at block transitions can be found in Chapter "Continuous Path Mode, Exact Stop and LookAhead (B1)".

Note

We recommend setting the following machine data for each axis with the same values: MD32431 MAX_AX_JERK and MD32432 PATH_TRANS_JERK_LIM (maximum axis-specific jerk for path movement at block transition)

5.3 Jerk limitation in JOG mode

The jerk limitation is active for axes in JOG mode during

- jogging
- handwheel jogging
- repositioning.

The jerk limitation is not active during

- reference point approach with
- alarms that initiate a rapid stop.

Jerk limitation can be determined for specific axes. The acceleration response corresponds with the SOFT acceleration profile of path-related jerk limitation. This limitation cannot be deselected for the axes in the relevant modes.

The axes for which jerk limitation is to be programmed can be selected with MD32420 JOG_AND_POS_JERK_ENABLE. The permissible axis-specific maximum jerk is stored in MD32430 JOG_AND_POS_MAX_JERK.

5.4 Data lists

Machine data

Number	Identifier	Name
Axis-specific		
32300	MAX_AX_ACCEL	Axis acceleration
32420	JOG_AND_POS_JERK_ENABLE	Enabling axis-specific jerk limitation
32430	JOG_AND_POS_MAX_JERK	Axis-specific jerk
32431	MAX_AX_JERK	Maximum axis-specific jerk during path movement
32432	PATH_TRANS_JERK_LIM	Maximum axis-specific jerk during path movement at the block transition

Manual Operation and Handwheel traversal 6

6.1 General characteristics of traversing in JOG

JOG mode

Axes/Spindles can be traversed manually in JOG mode. The active mode is transmitted to the PLC via the IS "Active mode: JOG" (DB3100.DBX0000.2) and is visible in the display, see also Chapter "Operating Modes, Program Operation (Page 85)".

Traversing possibilities

Traversing the axes can be done via the traverse keys of a connected machine control panel (manual travel) or via connected handwheels (handwheel jogging).

All machine axes can be traversed simultaneously using keys (with an appropriate version of a user-specific machine control panel) or via handwheel, depending on the number of handwheels connected. If several machine axes are moved simultaneously, there is no interpolatory relation.

Coordinate systems

The user has the option of traversing axes in the coordinate systems:

- Machine coordinate system (MCS); machine axes manually traversable
- Workpiece coordinate system (WCS); geometry axes manually traversable

Machine functions

Variants exist for **manual traverse** (the so-called machine functions):

- Continuous traversal
- Incremental traversing (INC, preset number of traversing increments). An increment is evaluated with 0.001 mm if the basic system setting is metric.

The PLC user program transfers a user-specific machine function queued at the machine control interface to the relevant PLC/NCK interface. Here the axis-specific NCK/PLC interface should be used for a machine axis/spindle, and the channel-specific NCK/PLC interface should be used for a geometry axis or valid for all axes/spindles and geometry axes: Signals in the operating mode range (see also following section).

Handwheel jogging

The axes can also be traversed via the handwheel in MCS or WCS. Incremental traversing (INC...) must be set to evaluate the handwheel pulses (see Section "Handwheel traversal in JOG (Page 63)").

Traversing the geometry axes

If workpieces whose workpiece coordinate system is not parallel to the machine coordinate system are being machined (inclined clamping, programmed rotation active in the contour), traversing can be done along the axes of the workpiece coordinate system via the traverse keys or handwheel. In the stopped state, switch from the operating mode "AUTO" to "JOG" and traverse a geometry axis instead of a machine axis. Depending on the active rotation of the workpiece coordinate system, between one and three machine axes move.

If a machine axis is traversed, this cannot also be moved via the traverse keys of a geometry axis. The traversing motion of the machine axis must first have been completed - otherwise alarm 20062 "Axis already active" is output. Two geometry axes can be traversed simultaneously with the handwheels 1 and 2.

Note

A separate, channel-specific PLC interface supplies geometry axes.

Transverse axis in "turning" technology

A geometry axis is defined as a transverse axis. If radius programming (DIAMOF) is selected instead of diameter programming (DIAMON), the following must be noted when traversing in JOG:

- Continuous traversing: There are no differences when a transverse axis is traversed continuously.
- Incremental traversing: Only half the distance of the selected increment size is traversed.
- Traversing with the handwheel: As for incremental travel, with the handwheel only half the distance is traversed per handwheel pulse.

Spindle manual travel

The spindle can also be traversed manually in the JOG mode. Essentially the same conditions apply as for manual traverse of machine axes. With JOG, the spindle can be traversed via the traverse keys/ IS "continuous" or "INC...". The mode is selected and activated via the axis-/spindle-specific PLC interface as for the axes.

Spindle manual travel is possible in positioning mode (spindle in position control) or in open-loop control mode. The parameter set (machine data) of the current gear stage applies.

Velocity

The velocity of the axes/spindle during manual traverse in JOG is defined by the following default values:

- For linear axes with the general SD41110 JOG_SET_VELO (JOG velocity with G94) or for rotary axes with SD41130 JOG_ROT_AX_SET_VELO (JOG velocity for rotary axes) or SD41200 JOG_SPIND_SET_VELO (JOG velocity for the spindle).
- If the corresponding SD is zero, the appropriate axis-specific MD32020 JOG_VELO (conventional axis velocity) applies. In this case, the value of the assigned machine axis is used for geometry axes: X->X1, Y->Y1, Z->Z1 (for default setting).

Rapid traverse override

If in the case of machine axes the rapid traverse override key is pressed at the same time as the traversing keys, then the movement is executed at the rapid traverse velocity set in axis-specific MD32010 JOG_VELO_RAPID (axis velocity in JOG mode with rapid traverse override).

The value of the assigned machine axis is used for geometry axes: X->X1, Y->Y1, Z->Z1 (for default setting). The separate PLC interface area of the geometry axes must be used for control.

Velocity override

The velocity at which axes traverse in JOG can also be influenced by the axis-specific feedrate override switch for machine axes, provided that axis-specific IS "Override active" (DB380x.DBX0001.7) is set. If the switch is set at 0%, the axis is not traversed - even if IS "Override active" is not set.

The channel-specific feedrate override switch applies to geometry axes, or, in the case of rapid traverse override, the rapid traverse override switch.

The activated spindle override switch applies to the spindle.

Acceleration

The maximum axis acceleration is defined with the axis-specific MD32300 MAX_AX_ACCEL. The acceleration can also be set via a preset characteristic curve in JOG mode. The possible settings are described in Chapter "Acceleration (Page 55)".

PLC interface

A separate PLC interface (DB3200.DBB1000, ff or DB3300.DBB1000, ff) exists for **geometry axes** (axes in WCS) that contains the same signals as the axis-specific PLC interface.

When the **spindle** is traversed manually, the PLC interface signals between the NCK and PLC have the same effect as for machine axes. Interface signals "Position reached with fine or coarse exact stop" are only set if the spindle is in position control.

In the case of interface signals that are only spindle-specific, while the spindle is traversing in JOG, the following should be noted:

- The following PLC interface signals to the spindle have no effect:
  - IS "Invert M3/M4" (DB380x.DBX2001.6)
  - IS "Set direction of rotation ccw" or "Set direction of rotation cw" (DB380x.DBX2002.7 or .6)
  - IS "Oscillation speed" (DB380x.DBX2001.5)
- The following PLC interface signals from the spindle are not set:
  - IS "Actual speed cw" (DB390x.DBX2001.7)
  - IS "Spindle in setpoint range" (DB390x.DBX2001.5)

Note

A reset causes the manual traverse motion (axis/spindle) to be terminated with brake ramp.

Limitations

The following limitations are active for manual travel:

- Software limit switches 1 or 2 (axis must be referenced)
- Hardware limit switches

The control ensures that the traversing movement is aborted as soon as the first valid limitation has been reached. Velocity control ensures that deceleration is initiated early enough for the axis to stop exactly at the limit position (e.g. software limit switch). Only when the hardware limit switch is triggered does the axis stop abruptly with "rapid stop".

An alarm is output when the corresponding limit is reached. The control automatically prevents further movement in this direction. The traversing keys and the handwheel have no effect in this direction.

Note

The software limit switches are only active if the axis has previously been referenced.

Note


The function for retracting an axis that has approached the limit position depends on the machine manufacturer. Please refer to the machine manufacturer's documentation!

For further information on working area limits and hardware and software limit switches see Chapter "Axis Monitoring (Page 31)".

6.2 Continuous travel

Selection

When JOG mode is selected, the active machine function "continuous" interface signal is set automatically:

- For geometry axes: DB3300.DBX1001.6, DB3300.DBX1005.6, DB3300.DBX1009.6
- For machine axes/spindle: DB390x.DBX0005.6

Continuous mode in JOG mode can also be selected via the PLC interface (IS "Machine function: continuous"). The PLC defines via the "INC inputs in mode group range active" interface signal (DB2600.DBX0001.0) the signal range within which INC/continuous signals are delivered to the NCK:

DB2600.DBX0001.0→ = 1	in the operating mode range: DB3000.DBB0002, valid for all axes
DB2600.DBX0001.0→ = 0	in the geometry axis / axis range: DB3200.DBB1001, DB3200.DBB1005, DB3200.DBB1009, DB380x.DBB0005

Traversing keys +/-

The plus and minus traversing keys are selected to move the relevant axis in the appropriate direction.

Traverse key signals PLC to NCK IS:

- For geometry axes (traverse in WCS):

  DB3200.DBX1000.7/.6, DB3200.DBX1004.7/.6, DB3200.DBX1008.7/.6
- For machine axes / spindle (traverse in MCS): DB80x.DBX004.7/.6

If both traversing keys of an axis are pressed simultaneously, there is no traversing movement, or, if an axis is in motion, it is stopped.

Motion command +/-

As soon as a traverse request for an axis/spindle is active (e.g. after selection of a traverse key), the IS "Travel command+" or "Travel command-" is sent to the PLC (depending on selected traverse direction):

- For geometry axes: DB3300.DBX1000.7/.6, DB3300.DBX1004.7/.6, DB3300.DBX1008.7/.6
- For machine axes / spindle: DB390x.DBX004.7 /.6

Continuous travel in jog mode

The axis traverses for as long as the traverse key is held down if no axis limit is reached first. When the traversing key is released, the axis is decelerated to standstill and the movement comes to an end.

6.3 Incremental travel (INC)

Programming increments

The path to be traversed by the axis is defined by so-called increments (also called "incremental dimensions"). The required increment must be set by the machine user before the axis is traversed.

The setting is made on the machine control panel, for example. After the corresponding logic operation, the IS "Machine function: INC1 to INCvar" associated with the required increment must be set by the PLC user program after it has been correctly linked. The PLC defines via the "INC inputs in mode group range active" interface signal (DB2600.DBX0001.0) the signal range within which INC signals are delivered to the NCK:

DB2600.DBX0001.0 = 1 →	in the operating mode range: DB3000.DBB0002, valid for all axes
DB2600.DBX0001.0 = 0 →	in the geometry axis / axis range: DB3200.DBB1001, DB3200.DBB1005, DB3200.DBB1009, DB380x.DBB0005

The active machine function IS "INC... " is signaled by the NCK to the PLC:

- For geometry axes: DB3300.DBX1001.0, DB3300.DBX1005.0, DB3300.DBX1009.0 to .5
- For machine axes / spindle: DB390x.DBX0005.0 to .5

Settable increments

The operator can set different increment sizes:

- **Fixed increments** whose increment sizes are common to all axes: INC1, INC10, INC100, INC1000 (only via IS: INC10000).
- **A variable increment** (INCvar). The increment setting for the variable increment can also be made for all axes using general SD41010 JOG_VAR_INCR_SIZE (size of the variable increment for INC/handwheel).

Traverse keys and travel command

As for continuous traversing (see Section "Continuous travel (Page 61)")

Abort traversing movement

If you do not want to traverse the whole increment, the traverse movement can be aborted with RESET or "Delete distance-to-go" interface signal (DB380x.DBX0002.2).

6.4 Handwheel traversal in JOG

Selection

JOG mode must be active. The user must also set the increment INC1, INC10, etc., which applies to handwheel travel.

Up to 2 handwheels can be connected. This means that up to 2 axes can be traversed by handwheel simultaneously and independently.

A handwheel is assigned to the geometry or machine axes (WCS or MCS) via interface signals.

The axis to be moved as a result of rotating handwheel 1 to 2 can be set:

- Via the PLC user interface with IS "Activate handwheel 1 to 2"
  - For machine axis (traverse in MCS): DB380x.DBX0004.0 to .2
  - For geometry axis (traverse in WCS): DB3200.DBX0000.0 to .2, DB3200.DBX0004.0 to .2, DB3200.DBX0008.0 to .2.

  The assignment is linked to the PLC interface through the PLC user program. Only here can several machine axes be assigned to one handwheel simultaneously.
- Using menu-assisted operation (HMI). Pressing the "Handwheel" softkey in the JOG-mode basic menu displays the "Handwheel" window. This enables an axis (WCS or MCS) to be assigned to each handwheel.

A separate user interface between the HMI and PLC is provided to allow activation of the handwheel from the operator panel (HMI). This interface that the basic PLC program supplies for handwheels 1 to 2 contains the following information:

- The axis numbers assigned to the handwheel IS "Axis number handwheel n"

  (DB1900.DBB1003, ff)
- Additional information on the machine or geometry axis

  IS "Machine axis" (DB1900.DBB1003.7, ff)

The "Activate handwheel" interface signal is either set to "0" (disable) or to "1" (enable) by the PLC user program for the defined axis.

Settings as path or velocity

When the electronic handwheel is turned, the assigned axis is traversed either in the positive or negative direction depending on the direction of rotation.

The general MD11346 HANDWH_TRUE_DISTANCE (handwheel path or velocity specification) can be used to set the setting type of the handwheel motion and thus adapted to the intended use.

- MD value = 0 (default):

  The settings from the handwheel are velocity specifications. When the handwheel is stationary, braking is realized along the shortest path.

- MD value = 1:

  The settings from the handwheel are path specifications. No pulses are lost. Limiting the velocity to the maximum permissible value can cause the axes to overtravel. Particular care should be taken in the case of a high weighting of the handwheel pulses. Further variants of the path or speed setting are possible with the value = 2 or 3.

Evaluation

The traversing path/velocity produced by rotation of the handwheel is dependent on the following factors:

- Number of handwheel pulses received at the interface
- Active increment (machine function INC1, INC10, INC100, ...)

  An increment is evaluated with 0.001 mm if the basic system setting is metric.
- Pulse weighting of the handwheel using general MD11320 HANDWH_IMP_PER_LATCH (handwheel pulses per locking position)

Motion command +/-

While the axis is moving, the "Travel command+" or "Travel command-" interface signal is transmitted to the PLC depending on the direction of motion.

- For geometry axes: DB3300.DBX1000.7/.6, DB3300.DBX1004.7/.6, DB3300.DBX1008.7/.6
- For machine axes / spindle: DB390x.DBX004.7/.6.

If the axis is already being moved using the traversing keys, the handwheel cannot be used. Alarm 20051 "Jogging with the handwheel not possible" is output.

Velocity

The velocity results from the pulses generated by the handwheel and the pulse evaluation: Traverse path per time unit. This velocity is limited by the value in the axis-specific MD32000 MAX_AX_VELO.

Abortion/interruption of traversing movement

The traversing movement is aborted as the result of a RESET or the axis-specific IS "Deletion of distance-to-go" (DB380x.DBX0002.2). The setpoint/actual-value difference is deleted.

NC STOP only interrupts the traversing movement. NC START releases the handwheel motion again.

Movement in the opposite direction

Depending on MD11310 HANDWH_REVERSE, the behavior when the traversing direction is reversed is as follows:

- MD value = 0:

  If the handwheel is moved in the opposite direction, the resulting distance is computed and the calculated end point is approached as fast as possible: If this end point is located before the point where the moving axis can decelerate in the current direction of travel, the unit is decelerated and the end point is approached by moving in the opposite direction. If this is not the case, the newly calculated end point is approached immediately.

- MD value > 0:

  If the handwheel is moved in the opposite direction by at least the number of pulses indicated in the machine data, the axis is decelerated as fast as possible and all pulses received until the end of interpolation are ignored. That means, another movement takes place only after standstill (setpoint side) of the axis (new function).

Response at software limit switches

When axes are traversed in JOG mode, they can traverse only up to the first active limitation before the corresponding alarm is output.

Depending on the machine data MD11310 HANDWH_REVERSE, the behavior is as follows (as long as the axis on the setpoint side has not yet reached the end point):

- MD value = 0:

  The distance resulting from the handwheel pulses forms a fictitious end point which is used for the subsequent calculations: If this fictitious end point is positioned, for example, 10 mm behind the limitation, these 10 mm must be traversed in the opposite direction before the axis traverses again. If a movement in the opposite direction is to be performed immediately after a limit, the fictitious distance-to-go can be deleted via IS "Delete distance-to-go" (DB380x.DBX0002.2) or by deselecting of the handwheel assignment.

- MD value > 0:

  All handwheel pulses leading to an end point behind the limitation are ignored. Any movement of the handwheel in the opposite direction leads to an immediate movement in the opposite direction, i.e. away from the limitation.

6.5 Fixed point approach in JOG

6.5.1 Introduction

Function

The machine user can use the "Approach fixed point in JOG" function to approach axis positions defined using machine data by actuating the traversing keys of the Machine Control Panel or by using the handwheel. The traveling axis comes to a standstill automatically on reaching the defined fixed point.

Applications

Typical applications are, for example:

- Approaching a basic position before starting an NC program.
- Travel towards tool change points, loading points and pallet change points.

Requirements

- The "Approaching fixed point in JOG" can be activated only in the "JOG" mode.

  The function cannot be enabled in the JOG-REPOS and JOG-REF sub-modes and in JOG in the "AUTO" mode.
- The axis to be traversed must be referenced.
- A kinematic transformation may not be active.
- The axis to be traversed must not be a synchronized axis of an active coupling.
- No ASUPs are executed.

Approaching a fixed point with G75

The process for approaching defined fixed points can be activated from the part program too using the `G75` command.

For more information on approaching fixed points with `G75`, refer to the SINUMERIK 808D Programming and Operating Manual, Section: "Fixed point approach".

6.5.2 Functionality

Procedure

Procedure in "Approaching fixed point in JOG"

- Selection of JOG mode
- Enabling the "Approach fixed point in JOG" function
- Traversing of the machine axis with traverse keys or handwheel

Activation

The PLC sets the interface signal after the "Approach fixed point in JOG" function is selected:
"JOG - Approach fixed point" (DB380x.DBX1001.0-2)

The number of the fixed point to be approached is output using bit 0 - 2 in binary code. The NC confirms activation with the following interface signal as soon as the function takes effect:
"JOG - Approaching fixed point active" (DB390x.DBX1001.0-2)

Sequence

The actual traversing is started with the traverse keys or the handwheel in the direction of the approaching fixed point.

The selected machine axis traverses till it comes to an automatic standstill at the fixed point.

The corresponding NC/PLC interface signal is sent on reaching the fixed point with "Exact stop fine":
"JOG - Approaching fixed point reached" (DB390x.DBX1001.3-5)

This display signal is also reported if the axis reaches the fixed point position in the machine coordinate system using other methods (e.g. NC program, synchronized action) at the setpoint end, and comes to a standstill at the actual-value end within the "Exact stop fine" tolerance window (MD36010 STOP_LIMIT_FINE)

Movement in the opposite direction

The response while traversing in the opposite direction (i.e. in the opposite direction to the one used when approaching the fixed point) depends on the setting of bit 2 in the following machine data:
MD10735 JOG_MODE_MASK (settings for JOG mode)

Traversing in the opposite direction is only possible if bit 2 is set.

Traversing in the opposite direction is blocked if bit 2 is not set, and the following channel status message is output if an attempt is made (using the traversing keys or the handwheel) to traverse in the opposite direction to the one used when approaching the fixed point:

"JOG: <Axis> direction blocked"

Approaching other fixed point

If a different fixed point is set during the fixed-point approach, the axis motion is stopped and the following alarm is signaled:
Alarm 17812 "Channel %1 axis %2 fixed-point approach in JOG: Fixed point changed"

The message signal "JOG - Approaching fixed point active" displays the number of the newly selected fixed point. The JOG traverse must be triggered again to continue traversing.

Note

To avoid the alarm message, the machine user should proceed as follows:

1. Cancel the current traverse movement with residual distance deletion.

2. Activate fixed point approach for another fixed point and start the operation after the axis comes to a standstill.

Withdrawal from fixed point / deactivation

To withdraw from a fixed position, you must deactivate the "Approaching fixed point in JOG" function. This is done by resetting the activation signal to "0".
DB380x.DBX1001.0-2 = 0

The message signals "JOG - Approaching fixed point active" and "JOG - Approaching fixed point reached" are canceled on leaving the fixed-point position.

Special case: Axis is already on fixed point

The axis cannot be moved if, while starting the fixed point traverse, the axis is already at the position of the fixed point to be approached. This is displayed through the following channel status message:
"JOG: <Axis> position reached"

To withdraw from the fixed position, you must deactivate the "Approaching fixed point in JOG" function.

Special features of incremental travel

If, during incremental travel, the fixed point is reached before the increment is completed, then the increment is considered to have been completed fully. This is the case even when only whole increments are traveled.

MD11346 HANDWH_TRUE_DISTANCE = 2 or 3

Features of modulo rotary axes

Modulo rotary axes can approach the fixed point in both directions (bit 2 of MD10735 has no significance for them). No attempt is made to follow the shortest path (DC) during the approach.

Features of spindles

A spindle changes to the positioning mode on actuating the "Approaching fixed point in JOG" function. The closed loop position control is active and the axis can traverse to the fixed point.

If a zero mark has not been detected, the following alarm message is output (as with axis operation):
Alarm 17810 "Channel %1 axis %2 not referenced"

As a spindle must also be a modulo rotary axis at all times, the same conditions apply for direction observation as for modulo rotary axes (refer to the paragraph "Features of modulo rotary axes")

6.5.3 Parameter setting

Movement in the opposite direction

The response while traversing in the opposite direction, i.e., against the direction of the approaching fixed point depends on the setting of Bit 2 in the machine data:

MD10735 JOG_MODE_MASK (settings for JOG mode)

Bit	Value	Description
2	0	Travel in the opposite direction is **not** possible (default setting).
	1	Movement in the opposite direction is possible.

Fixed point positions

A maximum of 4 fixed point positions can be defined for each axis via the following machine data:

MD30600 FIX_POINT_POS[n]

Number of valid fixed point positions

The number of valid fixed point positions of an axis is defined via the machine data:

MD30610 NUM_FIX_POINT_POS

Note

"Approaching fixed point with G75" constitutes an exception here. In this case, it is also possible to approach two fixed-point positions with one setting (MD30610 = 0).

6.5.4 Programming

System variables

The following system variables that can be read in the part program and in the synchronous actions for the "Approach fixed point" function.

System variable	Description
$AA_FIX_POINT_SELECTED [<Axis>]	Number of fixed point to be approached
$AA_FIX_POINT_ACT [<Axis>]	Number of the fixed point on which the axis is currently located

6.5.5 Supplementary Conditions

Axis is indexing axis

The axis is not traversed and an alarm is output if the axis to be traversed is an indexing axis and the fixed point position to be approached does not match an indexing position.

Frames active

All active frames are ignored. Traversing is performed in the machine coordinate system.

Offset values active

Active compensation values (external work offset, synchronized action offset $AA_OFF, online tool offset) are also applied. The fixed point is a position in the machine coordinates system.

An alarm is signaled if an offset movement (external work offset, synchronized action offset $AA_OFF, online tool offset) is made during a fixed-point approach in JOG. The position of the fixed point to be approached in the machine coordinates system is not reached; instead a position that would have been reached without active offset movement is reached. The NC/PLC interface signal "JOG - Approaching fixed point reached" (DB390x.DBX1001.3-5) is not signaled.

6.5.6 Application example

Target

A rotary axis (machine axis 4 [AX4]) is to be moved to Fixed Point 2 (90 degrees) with the "Approaching fixed point in JOG" function.

Parameter setting

The machine data for the "Approaching fixed point" function of machine axis 4 are parameterized as follows:

MD30610 NUM_FIX_POINT_POS[AX4] = 4	4 fixed points are defined for machine axis 4.
MD30600 FIX_POINT_POS[0,AX4] = 0	1st Fixed point of AX4 = 0 degree
MD30600 FIX_POINT_POS[1,AX4] = 90	2nd Fixed point of AX4 = 90 degree
MD30600 FIX_POINT_POS[2,AX4] = 180	3rd Fixed point of AX4 = 180 degree
MD30600 FIX_POINT_POS[3,AX4] = 270	4th Fixed point of AX4 = 270 degree

Initial situation

Machine axis 4 is referred and is in Position 0 degree. This corresponds to the 1st fixed point and is output via the following NC/PLC interface signal:
DB390x.DBX1001.0 = 1 (bit 0 - 2 = 1)

Approaching fixed point 2

The control system is switched in the JOG mode.

The "Approaching fixed point" procedure for fixed point 2 is activated via the following NC/PLC interface signal:
DB380x.DBX1002.1 = 1 (bit 0 - 2 = 2)

Activation is confirmed by the following NC/PLC interface signal:
DB390x.DBX1001.1 = 1 (bit 0 - 2 = 2)

The Plus traverse key in the machine control table is used to traverse continuously to approach Fixed Point 2.

The machine axis 4 stops at the 90 degree position. This is signaled via the following NC/PLC interface signal:
DB390x.DBX1001.4 = 1 (bit 3 - 5 = 2)

6.6 Data table

6.6.1 Machine data

Number	Identifier	Name
General information		
10000	AXCONF_MACHAX_NAME_TAB[n]	Machine axis name [n = axis index]
10735	JOG_MODE_MASK	Settings for JOG mode
11310	HANDWH_REVERSE	Defines movement in the opposite direction
11320	HANDWH_IMP_PER_LATCH[0]...[2]	Handwheel pulses per locking position
11346	HANDWH_TRUE_DISTANCE	Handwheel path or velocity specification
Channel-specific		
20060	AXCONF_GEOAX_NAME_TAB[n]	Geometry axis in channel [n = geometry axis index]
20100	DIAMETER_AX_DEF	Geometry axes with transverse axis functions
Axis/spindle-specific		
30600	FIX_POINT_POS[n]	Fixed-point positions for the axis
30610	NUM_FIX_POINT_POS	Number of fixed-point positions for an axis
32000	MAX_AX_VELO	Maximum axis velocity
32010	JOG_VELO_RAPID	Rapid traverse in JOG mode
32020	JOG_VELO	JOG axis velocity
32300	MAX_AX_ACCEL	Axis acceleration
32420	JOG_AND_POS_JERK_ENABLE	Enable for axis-spec. jerk limitation
32430	JOG_AND_POS_MAX_JERK	Axis-specific jerk
35130	GEAR_STEP_MAX_VELO_LIMIT[0]...[5]	Maximum velocity for gear stage/spindle

6.6.2 Setting data

Number	Identifier	Name
General information		
41010	JOG_VAR_INCR_SIZE	Size of variable increment for INC/handwheel
41110	JOG_SET_VELO	JOG velocity for linear axes
41130	JOG_ROT_AX_SET_VELO	JOG speed for rotary axes
41200	JOG_SPIND_SET_VELO	JOG velocity for the spindle

6.6.3 Interface signals

Number	Bit	Name
Signals from HMI to PLC		
DB1900.DBX1003	.0 to .2	Axis number for handwheel 1
DB1900.DBX1004	.0 to .2	Axis number for handwheel 2
NCK-specific		
DB2600.DBX0001	.0	INC inputs in operating mode range active
Specific to operating mode		
DB3000.DBX0000	.2	JOG mode
DB3000.DBX0002	.0 to .6	Machine function INC1 up to continuous in operating mode range
DB3100.DBX0000	.2	Active JOG mode
Channel-specific		
DB3200.DBX1000 DB3200.DBX1004 DB3200.DBX1008	.1, .0 .1, .0 .1, .0	Activate handwheel (2, 1) for geometry axis 1 for geometry axis 2 for geometry axis 3
DB3200.DBX1000 DB3200.DBX1004 DB3200.DBX1008	.4 .4 .4	Traversing-key lock for geometry axis 1 for geometry axis 2 for geometry axis 3
DB3200.DBX1000 DB3200.DBX1004 DB3200.DBX1008	.5 .5 .5	Rapid traverse override for geometry axis 1 for geometry axis 2 for geometry axis 3
DB3200.DBX1000 DB3200.DBX1004 DB3200.DBX1008	.7 or .6 .7 or .6 .7 or .6	Traversing keys plus or minus for geometry axis 1 for geometry axis 2 for geometry axis 3
DB3200.DBX1000 DB3200.DBX1004 DB3200.DBX1008	.0 to .6 .0 to .6 .0 to .6	Machine function INC1 to continuous for geometry axis 1 for geometry axis 2 for geometry axis 3
DB3300.DBX1000 DB3300.DBX1004 DB3300.DBX1008	.1, .0 .1, .0 .1, .0	Handwheel active (2, 1) for geometry axis 1 for geometry axis 2 for geometry axis 3
DB3300.DBX1000 DB3300.DBX1004 DB3300.DBX1008	.7 or .6 .7 or .6 .7 or .6	Traverse command plus or minus for geometry axis 1 for geometry axis 2 for geometry axis 3
DB3300.DBX1001 DB3300.DBX1005 DB3300.DBX1009	.0 to .6 .0 to .6 .0 to .6	Active machine function INC1 to continuous for geometry axis 1 for geometry axis 2 for geometry axis 3
Axis/spindle-specific		
DBB380x.DBX0000	-	Feed override
DB380x.DBX0000	.7	Override active
DB380x.DBX0002	.2	Delete distance-to-go
DB380x.DBX0004	.1, .0	Activate handwheel (2, 1)
DB380x.DBX0004	.4	Traversing-key lock
DB380x.DBX0004	.5	Rapid traverse override
DB380x.DBX0004	.7 or .6	Traversing keys plus or minus

Number	Bit	Name
DB380x.DBX0005	.0 to .6	Machine function INC1 up to continuous in axis range
DB380x.DBX1002	.0 to .2	Activated fixed-point approach in JOG (binary coded: fixed point 1 to 4)
DB390x.DBX0000	.7/.6	Position reached with coarse/fine exact stop
DB390x.DBX0004	.1, .0	Handwheel active (2, 1)
DB390x.DBX0004	.7 or .6	Traverse command plus or minus
DB390x.DBX0005	.0 to .6	Active machine function INC1 to continuous
DB390x.DBX1001	.0 to .2	Fixed-point approach in JOG active (binary coded)
DB390x.DBX1001	.3 to .5	Fixed point reached (binary coded)

Auxiliary function outputs to PLC

7

7.1 Brief description

Auxiliary functions

For the purpose of workpiece machining operations, it is possible to program process-related functions (feedrate, spindle speed or gear stages) and functions for controlling additional devices on the machine tool (sleeve forward, gripper open, clamp chuck) in the part program in addition to axis positions and interpolation methods. This is performed with the "auxiliary functions" as collective term for various types.

The following types of auxiliary functions are available:

- Miscellaneous function M
- Spindle function (S)
- Auxiliary function (H)
- Tool number T
- Tool offset D
- Feed F (for SINUMERIK 808D there is no output from F to PLC)

Output of auxiliary functions to PLC

The auxiliary function output sends information to the PLC indicating, for example, when the NC program needs the PLC to perform specific switching operations on the machine tool. The auxiliary functions are output, together with their parameters, to the PLC.

The values and signals must be processed by the PLC user program. The following section describes the various methods of configuring and programming auxiliary functions as well as their operating principles.

Auxiliary function groups

Auxiliary functions can be combined to form groups.

7.2 Programming of auxiliary functions

General structure of an auxiliary function

Letter[address extension]=Value

The letters which can be used for auxiliary functions are: **M, S, H, T, D, F.**

The address extension must be an integer. The square brackets can be omitted when an address extension is specified directly as a numeric value.

The value is defined differently for the individual auxiliary functions:

- INT= integer
- REAL= fractional decimal number (floating point)

Table 7- 1 Overview of auxiliary functions, programming

Function	Address extension (integer)		Value			Explanation	Number per block
	Meaning	Area	Area	Type	Meaning		max
M	Spindle no.	1 - 2	0-99	INT	Function	Specific numbers are assigned a fixed function.	5
S	Spindle no.	1 - 2	0-±3.4028 ex 38	REAL	Spindle speed		1
H	Any	0 - 99	±3.4028 ex 38	REAL	Any	Functions have no effect in the NCK; only to be implemented on the PLC	3
T	-	-	0-32000	INT	Tool selection		1
D	-	-	0-9	INT	Tool offset selection	D0 deselection, default D1	1
F	-	-	0,001- 999 999,999	REAL	Path feedrate		1

A maximum total of 10 auxiliary functions may be programmed in one block. Alarm 14770 "Auxiliary function incorrectly programmed" is output when the specified length for address extension of value is exceeded or when the wrong data type is used. The following table shows some programming examples for H functions.

If the admissible number of auxiliary functions per block is exceeded, alarm 12010 is issued.

Table 7- 2 Programming examples of H functions

Programming	Output of H function to the PLC
H5	H0=5.0
H=5.379	H0=5.379
H17=3.5	H17=3.5
H5.3=21	Error, alarm 14770

Block change

A new auxiliary function output from the NCK to the PLC is only possible after the PLC has acknowledged all transferred auxiliary functions. Auxiliary functions are present in the user interface for at least one PLC cycle. A block is considered as completed when the programmed movement has been completed and the auxiliary function has been acknowledged. To do so, the NCK stops the part program processing if necessary to ensure that no auxiliary functions are lost from the PLC user program's point of view.

7.3 Transfer of values and signals to the PLC interface

Time of transfer

In the case of auxiliary functions which are output at the end of a block (e.g. M2), the output is only made after all axis movements and the SPOS movement of the spindle have been completed.

If several auxiliary functions with different output types (prior, during, at end of motion) are programmed in one motion block, then they are output individually according to their output type.

In a block without axis movements or SPOS movement of the spindle, the auxiliary functions are all output immediately in a block.

Continuous-path mode

A path movement can only remain continuous if auxiliary function output takes place **during the movement** and is acknowledged by the PLC before the path end is reached, see Chapter "Continuous Path Mode (Page 48)".

Interface signals

Transfer of the signals from NCK to the PLC.

7.4 Grouping of auxiliary functions

Functionality

The auxiliary functions of the types M, H, D, T, and S that are to be issued can be grouped to auxiliary function groups through the machine data.

An auxiliary function can only be assigned to one group.

Only one auxiliary function of a group can be programmed per block. Otherwise, alarm 14760 is issued.

Configuration

You can define a maximum of 64 auxiliary function groups. A maximum of 64 auxiliary functions can be assigned to these 64 auxiliary function groups. This number does not include auxiliary functions (group 1 to 3) that are pre-assigned as standard.

The actual number of auxiliary functions that are to be assigned must be entered in the NCK-specific MD11100 AUXFU_MAXNUM_GROUP_ASSIGN (number of the auxiliary functions distributed to the AUXFU groups). To do so, the password for protection level 2 must be set. Then, the control must be turned off and on again. Now, the subsequent machine data with an index n greater than zero are available and additional values can be entered.

An allocated auxiliary function is defined in the following machine data:

- MD22000 AUXFU_ASSIGN_GROUP[n] (auxiliary function group)
- MD22010 AUXFU_ASSIGN_TYPE[n] (auxiliary function type)
- MD22020 AUXFU_ASSIGN_EXTENSION[n] (auxiliary function extension)
- MD22030 AUXFU_ASSIGN_VALUE[n] (auxiliary function value)

Predefined auxiliary function groups

Group 1:

The auxiliary functions M0, M1, and M2 (M17, M30) are, by default, allocated to group 1. The output is always made at the end of the block.

Group 2:

The M functions M3, M4, and M5 (M70) are, by default, allocated to group 2. The output is always made before the movement.

Group 3:

The S function is, by default, contained in group 3. The output is made with the movement.

User-defined groups

The other (user-defined) groups are issued with the movement.

Ungrouped auxiliary functions

The output of auxiliary functions that are not assigned to groups is made with the movement.

Configuring example:

Distribute 8 auxiliary functions to 7 groups:

Group 1: M0, M1, M2 (M17, M30) - by default, should be kept

Group 2: M3, M4, M5 (M70) - by default, should be kept

Group 3: S functions - by default, should be kept

Group 4: M78, M79

Group 5: M80, M81

Group 6: H1=10, H1=11, H1=12

Group 7: all T functions

Password for protection level 2 is set.

Make entry in MD11100 AUXFU_MAXNUM_GROUP_ASSIGN=8.

Then turn off the control and turn it on again or perform the control start-up through the softkey and define the remaining machine data with a subsequent restart of the control.

Table 7- 3 Entries into the machine data for the example

Index n	MD22000 (GROUP)	MD22010 (TYPE)	MD22020 (EXTENSION)	MD22030 (VALUE)
0	4	M	0	78
1	4	M	0	79
2	5	M	0	80
3	5	M	0	81
4	6	H	1	10
5	6	H	1	11
6	6	H	1	12
7	7	T	0	-1

7.5 Block-search response

Block search with calculation

For the block search with calculation all auxiliary functions that are assigned to a group are collected and are issued at the end of the block search before the actual re-entry block (except for group 1: M0, M1,...). The last auxiliary function of a group is issued.

All collected auxiliary functions are issued in a separate block as regular auxiliary functions and before the movement.

Note

If the auxiliary functions are to be collected during the block search, they must be assigned to an auxiliary function group!

7.6 Description of auxiliary functions

7.6.1 M function

Application

You can use the M functions to enable the various switching operations on the machine per part program.

Scope of functions

- Five M functions per part program block are possible.
- Value range of M functions: 0 to 99; integer number
- Permanent functions have already been assigned to some of the M functions by the control manufacturer (see the Programming and Operating Manual). The functions not yet assigned fixed functions are reserved for free use of the machine manufacturer.

7.6.2 T function

Application

The T function can be used to make the tool required for a machining operation available through the PLC. Whether a tool change is to be performed directly with the T command or with a subsequent M6 command can be set in MD22550 TOOL_CHANGE_MODE.

The programmed T function can be interpreted as tool number or as location number.

Scope of functions

One T function per part program block is possible.

Peculiarity

T0 is reserved for the following function: remove the current tool from the tool holder without loading a new tool.

7.6.3 D function

The D function is used to select the tool offset for the active tool. Tool offsets are described in detail under:

Reference:

Programming and Operating Manual

7.6.4 H function

Application

The H functions can be used to transfer different values from the part program to the PLC. The meaning can be chosen by the user.

Scope of functions

- Three H functions per part program block are possible.
- Value range of the H functions: Floating data (as calculating parameter R)
- Address extension 0 to 99 (H0=... to H99=...) possible

7.6.5 S function

The S function is used to determine the speed for the spindle with M3 or M4. For turning machines with G96 (constant cutting speed) the cutting value is specified.

Reference:

Programming and Operating Manual

7.7 Data table

7.7.1 Machine data

Number	Identifier	Name
General		
11100	AUXFU_MAXNUM_GROUP_ASSIGN	Number of auxiliary functions distributed among the AUXFU groups
Channel-specific		
22000	AUXFU_ASSIGN_GROUP[n]	Auxiliary function groups
22010	AUXFU_ASSIGN_TYPE[n]	Auxiliary function types
22020	AUXFU_ASSIGN_EXTENSION[n]	Auxiliary function extensions
22030	AUXFU_ASSIGN_VALUE[n]	Auxiliary function values

7.7.2 Interface signals

Number	Bit	Name
Channel-specific		
DB2500.DBX0000	.0 to .4	M function 1 change to M function 5 change
DB2500.DBX0006	.0	S function 1 change
DB2500.DBX0008	.0	T function 1 change
DB2500.DBX0010	.0	D function 1 change
DB2500.DBX0012	.0 to .2	H function 1 change to H function 3 change
DB2500.DBD2000		T function 1 (DINT)
DB2500.DBD3000		M function 1 (DINT)
DB2500.DBB3004		Extended address of M function 1 (BYTE)
DB2500.DBD3008		M function 2 (DINT)
DB2500.DBB3012		Extended address of M function 2 (BYTE)
DB2500.DBD3016		M function 3 (DINT)
DB2500.DBB3020		Extended address of M function 3 (BYTE)
DB2500.DBD3024		M function 4 (DINT)
DB2500.DBB3028		Extended address of M function 4 (BYTE)
DB2500.DBD3032		M function 5 (DINT)
DB2500.DBB3036		Extended address of M function 5 (BYTE)
DB2500.DBD4000		S function 1 (REAL format)
DB2500.DBB4004		Extended address of S function 1 (BYTE)
DB2500.DBD4008		S function 2 (REAL format)
DB2500.DBB4012		Extended address of S function 2 (BYTE)
DB2500.DBD5000		D function 1 (DINT)

Number	Bit	Name
DB2500.DBW6004		Extended address of H function 1 (Word)
DB2500.DBD6000		H function 1 (REAL format)
DB2500.DBW6012		Extended address of H function 2 (Word)
DB2500.DBD6008		H function 2 (REAL format)
DB2500.DBW6020		Extended address of H function 3 (Word)
DB2500.DBD6016		H function 3 (REAL format)
DB2500.DBX1000	.0 - .7	Decoded M signals: M00 - M07
DB2500.DBX1001	.0 - .7	Decoded M signals: M08 - M15
DB2500.DBX1012	.0 - .7	Decoded M signals: M96 - M99
DB370x.DBD0000	-	M function for the spindle (DINT), axis-specific
DB370x.DBD0004	-	S function for the spindle (REAL), axis-specific

Operating Modes, Program Operation 8

8.1 Brief description

Program operation

The execution of part programs or part program blocks in "AUTO" or "MDA" modes is referred to as program operation. During execution, the program sequence can be controlled by PLC interface signals and commands.

Channel

A channel constitutes a unit in which a part program can be executed.

A channel is assigned an interpolator with program processing by the system. A certain mode is valid for it.

The SINUMERIK 808D control system has one channel.

8.2 Operating modes

8.2.1 Operating modes

Activating

The required operating mode is activated by the interface signals in the DB3000.DBB0000. If several modes are selected at the same time the priority of the operating modes is as follows:

- JOG (high priority): The axes can be traversed manually with the handwheel or the traversing keys. Channel-specific signals and interlocks are not observed.
- MDA: Program blocks can be processed
- AUTO (lower priority): Automatic processing of part programs

Feedback signal

The active operating mode is displayed by the interface signals in the DB3100.DBB0000.

Possible machine functions in JOG

The following machine function can be selected in the "JOG" operating mode:
REF (reference point approach)

The required machine function is activated with IS "REF" (DB3000.DBX0001.2). The display is visible in the IS "active machine function REF" (DB3100.DBX0001.2).

Possible machine functions in MDA

The following machine function can be selected in the "MDA" operating mode:
TEACH IN (insert program blocks)

The required machine function is activated with IS "TEACH IN" (DB3000.DBX0001.0). The display is visible in the IS "Active machine function TEACH IN" (DB3100.DBX0001.0).

Stop

A stop signal can be issued with the following interface signals

- IS "NC stop" (DB3200.DBX0007.3)
- IS "NC stop axes plus spindles" (DB3200.DBX0007.4)
- IS "NC stop at block limit" (DB3200.DBX0007.2)

Depending on the interface signal used, either only the axes or in addition the spindles of the channels are stopped or the axes at block end.

RESET

The active part program is aborted by the IS "Reset" (DB3000.DBX0000.7).

The following actions are executed when the IS "Reset" is triggered:

- Part program preparation is stopped immediately.
- Axes and spindles are stopped.
- Any auxiliary functions of the current block not yet output, are no longer output.
- The block indicator is reset to the beginning of the relevant part program.
- All Reset alarms are deleted from the display.
- The reset is complete as soon as IS "Channel status Reset" (DB3300.DBX0003.7) is set.

Ready

Ready to run is displayed by IS "808D Ready" (DB3100.DBX0000.3).

8.2.2 Mode change

General

A changeover to another operating mode is requested and activated via the interface.

Note

The mode is not changed internally until the IS "Channel status active" (DB3300.DBX0003.5) is **no longer** present.

In the "Channel status Reset" (IS: DB3300.DBX0003.7, e.g. after pressing the "Reset key") one can switch from any operating mode into another.

In the "Channel status interrupted" (IS: DB3300.DBX0003.6) only a conditional changeover is possible (see following table).

If one leaves AUTO to change to JOG, one must return to AUTO again or press "Reset". Thus a change AUTO-JOG-MDA is made impossible. The same applies for MDA from which one may change neither directly nor indirectly to AUTO, provided the Reset state is present.

The table shows the possible operating mode changes depending on the current operating mode and the channel state ("Channel in reset" or "Channel interrupted").

Table 8- 1 Operating mode change depending on channel state

	From	AUTO		JOG			MDI	
					AUTO previously	MDI previously		
To		Reset	Interrupt	Reset	Interrupt	Interrupt	Reset	Interrupt
AUTO				X	X		X	
JOG		X	X				X	X
MDI		X		X		X		

Possible mode changes are shown by an "X".

Error on operating mode changeover

A corresponding error message is output if a mode change request is rejected by the system. This error message can be cleared without changing the channel status.

Mode change disable

Changeover between operating modes can be inhibited by means of IS "Mode group changeover disable" (DB3000.DBX000.4). This suppresses the mode change request.

8.2.3 Functional possibilities in the individual modes

Overview of the functions

You see from the following table which function can be selected in which operating mode and in which operating state.

Table 8- 2 Functional possibilities in the individual modes

Mode of operation	**AUTO**			**JOG**						**MDI**				
Functions	**1**	**2**	**3**	**1**	**3**	**4**	**3**	**5**	**3**	**1**	**2**	**3**	**6**	**7**
Loading a part program from outside through "Services"	**sb**	**sb**		**sb**		**sb**		**sb**	**sb**	**sb**	**sb**			
Processing a part program/block	s	s	b							s	s	b		
Block search	s	s	b											
Reference point approach via part program command (G74)			**sb**									**sb**		

s: Function can be started in this status
b: Function can be processed in this status

1: Channel in reset
2: Channel interrupted
3: Channel active
4: Channel interrupted JOG during AUTO interruption
5: Channel interrupted JOG during MDA interruption
6: Channel active JOG in MDA during MDA interruption
7: Channel active JOG in MDA

8.2.4 Monitoring functions in the individual modes

Overview of monitoring functions

Different monitoring functions are active in individual operating modes.

Table 8- 3 Monitoring functions and interlocks

Mode of operation	AUTO			JOG						MDI				
Functions	1	2	3	1	3	4	3	5	3	1	2	3	6	7
Axis-specific monitoring functions or when positioning the spindle														
SW limit switch +			x		x		x		x			x	x	x
SW limit switch –			x		x		x		x			x	x	x
HW limit switch +	x	x	x	x	x	x	x	x	x	x	x	x	x	x
HW limit switch –	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Exact stop coarse/fine	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Clamping tolerance	x	x	x	x	x	x	x	x	x	x	x	x	x	x
DAC limit (analog spindle)	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Contour monitoring			x		x		x		x			x	x	x
Spindle-specific monitoring functions														
Speed limit exceeded			x		x		x		x			x		x
Spindle is stationary	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Spindle synchronized			x		x		x		x			x		x
Speed in setpoint range			x											
Maximum permissible speed			x		x		x		x			x		x
Encoder frequency limit			x		x		x		x			x		x

x: Monitoring is active in this status

1: Channel in reset
2: Channel interrupted
3: Channel active
4: Channel interrupted JOG during AUTO interruption
5: Channel interrupted JOG during MDA interruption
6: Channel active JOG in MDA during MDA interruption
7: Channel active JOG in MDA

8.2.5 Interlocks in the individual modes

Overview of interlocks

Different interlocks can be active in the different operating modes.

The following table shows which interlocks can be activated in which operating mode and in which operating state.

Mode of operation	AUTO			JOG						MDI				
Functions	**1**	**2**	**3**	**1**	**3**	**4**	**3**	**5**	**3**	**1**	**2**	**3**	**6**	**7**
General interlocks														
808D Ready	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Mode change disable	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Channel-specific interlocks														
Feed stop			x		x		x		x			x	x	x
NC Start disable	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Read-in disable	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Axis-specific interlocks														
Spindle disable	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Controller disable	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Axis disable	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Spindle-specific interlocks														
Controller disable	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Spindle disable	x	x	x	x	x	x	x	x	x	x	x	x	x	x

x: Interlock can be activated in this status

1: Channel in reset
2: Channel interrupted
3: Channel active
4: Channel interrupted JOG during AUTO interruption
5: Channel interrupted JOG during MDA interruption
6: Channel active JOG in MDA during MDA interruption
7: Channel active JOG in MDA

8.3 Processing a part program

8.3.1 Program mode and part program selection

Definition

Program mode applies if a part program is processed in the "AUTO" mode or program blocks are processed in the "MDA" mode.

Channel control

The Program mode can be controlled even while being executed via interface signals from the PLC. These can be either mode group specific or channel specific interface signals.

The channel reports its current program operation status to the PLC with interface signals.

Selection

A part program can be selected only if the relevant channel is in the Reset state.

The part program can be selected via:

- operator input ("MACHINE" operating area) / Program manager
- the PLC
  - Selection of a program via the program number in "Program list" (see the Programming and Operating Manual)
  - Reselection of an active program via the PLC-HMI interface (see Section "Signals from HMI to PLC (Page 21)")

8.3.2 Start of part program or part program block

START command, channel status

The channel-specific IS "NC start" (DB3200.DBX0007.1), which is usually controlled from the machine control panel key <CYCLE START>, starts program processing.

The START command can only be executed in the "AUTO" and "MDA" modes. For this purpose, the channel must be in the "Channel status reset" (DB3300.DBX0003.7) or "Channel status interrupted" (DB3300.DBX0003.6).

Required signal states

The selected part program can now be enabled for processing with the START command. The following enable signals are relevant:

IS "808D Ready" (DB3100.DBX0000.3)	must be set
IS "Activate program test" (DB3200.DBX0001.7)	may not be set
IS "NC Start disable" (DB3200.DBX0007.0)	may not be set
IS "NC Stop at block limit" (DB3200.DBX0007.2)	may not be set
IS "NC stop" (DB3200.DBX0007.3)	may not be set
IS "NC Stop axes plus spindle" (DB3200.DBX0007.4)	may not be set
IS "EMERGENCY STOP" (DB2700.DBX0000.1)	may not be set
Axis or NCK alarm	may not be present

Execution of command

The part program or part program block is automatically processed and IS "Channel status active" (DB3300.DBX0003.5) and IS "Program status running" (DB3300.DBX0003.0) are set.

The program is processed until the end of the program has been reached or the channel is interrupted or aborted by a STOP or RESET command.

Interrupts

The START command is not effective if the prerequisite is not fulfilled. Then one of the following interrupts occurs: 10200, 10202, 10203

8.3.3 Part program interruption

Channel status

The STOP command is executed only if the channel concerned is in the "Channel active" status (DB3300.DBX0003.5).

STOP commands

There are various commands which stop processing of the program and set the channel status to "interrupted":

- IS "NC Stop at block limit" (DB3200.DBX0007.2)
- IS "NC stop" (DB3200.DBX0007.3)
- IS "NC Stop axes plus spindle" (DB3200.DBX0007.4)
- IS "Single block" (DB3200.DBX0000.4)
- Programming command "M0" or "M1" and corresponding activation

Execution of command

After execution of the STOP command, IS "Program status stopped" (DB3300.DBX0003.2) and the IS "Channel status interrupted" (DB3300.DBX0003.6) are set. Processing of the interrupted part program can continue from the point of interruption with another START command.

The following actions are executed when the STOP command is triggered:

- Part program processing is stopped at the next block limit (with NC stop at block limit, M0/M1 or single block), processing is stopped immediately with the other STOP commands.
- Any auxiliary functions of the current block not yet output, are no longer output.
- The axes are stopped with subsequent stop of the part program processing.
- The block indicator stops at the point of interruption.

8.3.4 RESET command

Function

The RESET command (IS "Reset" (DB3000.DBX000.7)) can be executed in every channel state. This command is aborted by another command.

A RESET command can be used to interrupt an active part program or part program blocks. After execution of the Reset command, IS "Channel status reset" (DB3300.DBX0003.7) and the IS "Program status aborted" (DB3300.DBX0003.4) are set.

The part program cannot be continued at the point of interruption. All axes in the channel are at exact stop.

The following actions are executed when the RESET command is triggered:

- Part program preparation is stopped immediately.
- All axes and if appropriate spindles are braked.
- Any auxiliary functions of the current block not yet output, are no longer output.
- The block indicator is reset to the beginning of the part program.
- All alarms are cleared from the display if they are not POWER ON alarms.

8.3.5 Program control

Selection/activation

The user can control part program processing via the user interface. Under the "Program control" menu (operating mode "AUTO", operating area "MACHINE") certain functions can be selected, whereby some functions act on interface signals of the PLC. These signals are merely selection signals from the user interface. They do not activate the selected function.

These signal states must be transferred from the PLC user program to another area of the data block to activate the selected functions. With program control by the PLC the signals are to be set directly.

Table 8- 4 Program control

Function	Selection signal	Activation signal	Checkback signal
SKP skip block	DB1700.DBX0001.0	DB3200.DBX0002.0	
DRY dry run feedrate	DB1700.DBX0000.6	DB3200.DBX0000.6	
ROV rapid traverse override	DB1700.DBX0001.3	DB3200.DBX0006.6	
Preselection: SBL -single block coarse SBL -single block fine Single block	 - - User-specific	 - - DB3200.DBX0000.4	
M1 programmed stop	DB1700.DBX0000.5	DB3200.DBX0000.5	DB3300.DBX0000.5
PRT program test	DB1700.DBX0000.7	DB3200.DBX0001.7	DB3300.DBX0001.7

8.3.6 Program status

Program states

The status of the selected program is displayed in the interface in the "AUTO" and "MDA" operating modes. If the "JOG" operating mode is selected when the program is stopped, then the "interrupted" program status is displayed there or on reset also "aborted".

The following program states are available in SINUMERIK 808D:

- IS "Program status aborted" (DB3300.DBX0003.4)
- IS "Program status interrupted" (DB3300.DBX0003.3)
- IS "Program status stopped" (DB3300.DBX0003.2)
- IS "Program status running" (DB3300.DBX0003.0)

The effect of commands/signals

The program status can be controlled by activating different commands or interface signals. The following table shows the resulting program state when these signals are set (status before the signal is set -> Program status running).

Table 8- 5 Effect on program status

Commands	Program execution states			
	Aborted	Interrupted	Stopped	Running
IS "Reset"	X			
IS "NC Stop"			X	
IS "NC stop at block limit"			X	
IS "NC stop axes and spindles"			X	
IS "Read-in disable"				X
IS "Feed stop, channel-sp."				X
IS "Feed stop, axis-sp."				X
Feed override = 0%				X
IS "Spindle stop"				X
M2 in the block	X			
M0/M1 in the block			X	
IS "Single block"			X	
Auxiliary functions output to PLC but not yet acknowledged			X	

8.3.7 Channel status

Channel states

The current channel status is signaled at the interface for the channel. The PLC can then trigger certain responses and interlocks configured by the manufacturer depending on the status at the interface. The channel status is displayed in all operating modes.

The following channel states are available:

- IS "Channel status reset" (DB3300.DBX0003.7)
- IS "Channel status interrupted" (DB3300.DBX0003.6)
- IS "Channel status active" (DB3300.DBX0003.5)

The effect of commands/signals

The channel status can be modified through the activation of various commands or interface signals. The following table shows the resulting channel status when these signals are set (assumed status before the signal is set - > Channel status active).

The "Channel status active" signal is obtained when a part program or part program block is being executed or when the axes are traversed in JOG mode.

Table 8- 6 Effect on channel status

Commands	Resulting channel status		
	Reset	Interrupted	Active
IS "Reset"	X		
IS "NC Stop"		X	
IS "NC stop at block limit"		X	
IS "NC stop axes and spindles"		X	
IS "Read-in disable"			X
IS "Feed stop, channel-sp."			X
IS "Feed stop, axis-sp."			X
Feed override = 0 %			
IS "Spindle stop"			X
M2 in the block	X		
M0/M1 in the block		X	
IS "Single block"		X	
Auxiliary functions output to PLC but not yet acknowledged			X

8.3.8 Event-driven program calls

Application

In the case of certain events, an implied user program is to start. This allows the user to activate the initial settings of functions or carry out initialization routines by part program command.

Event selection

MD20108 PROG_EVENT_MASK (event-driven program call) can be used to specify which of the following events is to enable the user program:

- Bit0 = 1: Part program start
- Bit1 = 1: Part program end
- Bit2 = 1: Operator panel reset
- Bit3 = 1: Power up (of the NC control)

The user program is stored per default under the path name **/_N_CMA_DIR/_N_PROG_EVENT_SPF**. Another program name can be specified in MD11620 PROG_EVENT_NAME.

Other program name

A name is specified in MD11620 PROG_EVENT_NAME (program name of PROG_EVENT). The following directories are searched for the user program in the specified sequence:

- /_N_CUS_DIR/ for user cycles
- /_N_CMA_DIR/ for manufacturer cycles

The first program found with the stored name is called when a configured event occurs.

The same protection mechanisms that can be activated for cycles (protection levels for writing, reading etc.) are activated.

MD20108 PROG_EVENT_MASK is ignored during the simulation.

Request which start event

In the user program, the system variable $P_PROG_EVENT can be used to request the event, which enabled the part program.

Event

Part program start

Table 8- 7 Sequence when starting a part program

Sequence	Command	Boundary conditions (must be satisfied before the command)	Comments
1	Channel selection: Reset status Operating mode selection: AUTO or AUTO and overstoring or MDA or TEACH IN	None	Select channel and mode
2	NC Start	None	NCK start
3	MD20112 START_MODE_MASK	Initialization sequence with evaluation	
4	/_N_CMA_DIR/_N_PROG_EVENT_SPF or name from MD11620	as a subroutine	Implied call of the path name as a subroutine
5		None	Processing of the data part of the main program
6		None	Processing of the program part of the main program

Event

Part program end

Table 8- 8 Sequence at part program end

Sequence	Command	Boundary conditions (must be satisfied before the command)	Comments
1	Channel selection: Reset status Operating mode selection: AUTO or AUTO and overstoring or MDA or TEACH IN	None	Select channel and mode
2	NC Start	Block with end of part program	Block is changed
3	MD20110 RESET_MODE_MASK, MD20150 GCODE_RESET_VALUES, MD20152 GCODE_RESET_MODE	Control activated: Reset sequence with evaluation	
4	/_N_CMA_DIR/_N_PROG_EVENT_SPF or name from MD11620	as an ASUP	Implied call of the path name as an ASUP
5	MD20110 RESET_MODE_MASK, MD20150 GCODE_RESET_VALUES, MD20152 GCODE_RESET_MODE	Control activated: Reset sequence with evaluation	The G code reset position continues to be specified with machine data

Event

Operator panel reset

Table 8- 9 Processing sequence in operator panel reset

Sequence	Command	Boundary conditions (must be satisfied before the command)	Comments
1	Selection of channel and mode: any	Initial state: Any mode, any channel status	Select mode / channel status from any state
2	Reset		
3	MD20110 RESET_MODE_MASK, MD20150 GCODE_RESET_VALUES, MD20152 GCODE_RESET_MODE	Control activated: Reset sequence with evaluation	
4	/_N_CMA_DIR/_N_PROG_EVENT_SPF or name from MD11620	as an ASUP	Implied call of the path name as an ASUP
5	MD20110 RESET_MODE_MASK, MD20150 GCODE_RESET_VALUES, MD20152 GCODE_RESET_MODE	Control activated: Reset sequence with evaluation	The G code reset position continues to be specified with machine data

Event

Startup

Table 8- 10 Sequence with Powerup

Sequence	Command	Boundary conditions (must be satisfied before the command)	Comments
1	Reset	after power up	
2	MD20110 RESET_MODE_MASK, MD20150 GCODE_RESET_VALUES, MD20152 GCODE_RESET_MODE	Control activated after ramp up: Reset sequence with evaluation	
3	/_N_CMA_DIR/_N_PROG_EVENT_SPF or name from MD11620	as an ASUP	Implied call of the path name as an ASUP
4	MD20110 RESET_MODE_MASK, MD20150 GCODE_RESET_VALUES, MD20152 GCODE_RESET_MODE	Control activated: Reset sequence with evaluation	The G code reset position continues to be specified with machine data

Chronological sequences

For part program start and part program end:

Time sequence of VDI signals DB3300.DBB0003 ("Program status" and "Channel status") when processing a part program with an event-driven program call for part program start and part program end:

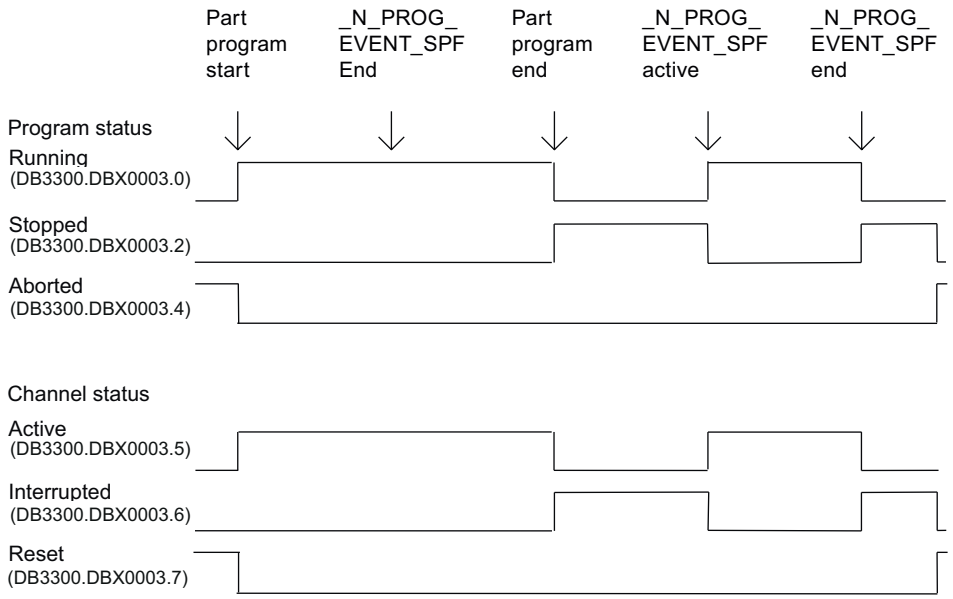


Figure 8-1 Time sequence of the interface signals for program status and channel status (1)

With operator panel reset:

Time sequence of VDI signals DB3300.DBB0003 ("Program status" and "Channel status") when processing with an event-driven program call:

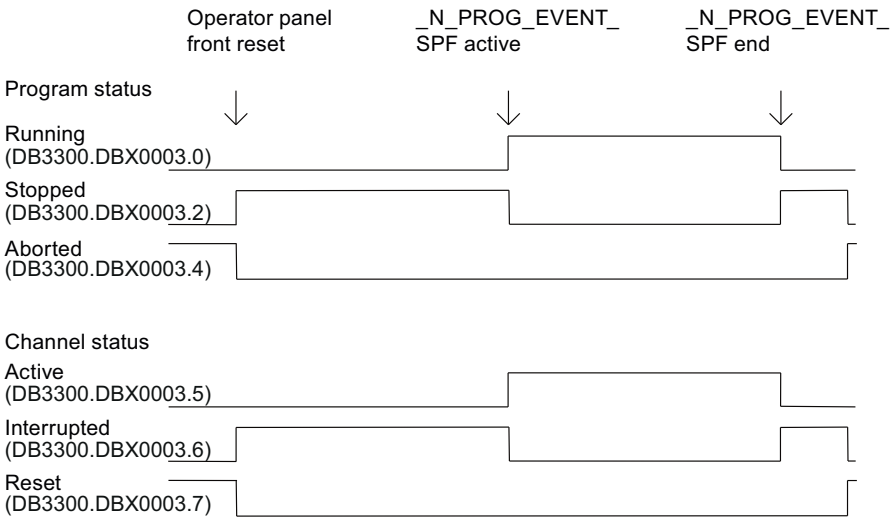


Figure 8-2 Time sequence of the interface signals for program status and channel status (2)

Note

IS DB3300.DBX0003.4 ("Program status aborted") and DB3300.DBX0003.7 ("Channel status reset") are only received if _N_PROG_EVENT_SPF has been completed.

Neither IS DB3300.DBX0003.4 ("Program status aborted") nor DB3300.DBX0003.7 ("Channel status reset") are received between the program end and the start of the program event.

This is also the case between an operator panel reset and the start of the program event.

Special points to be noted

The following must be noted for user program _N_PROG_EVENT_SPF:

- It is run with the lowest priority and can, therefore, be interrupted by the user ASUP.
- The PLC can be advised of the processing status of _N_PROG_EVENT_SPF via user M functions.
- The triggering event can be defined at the **interface** via the PLC program:

  DB3300.DBB4004 offers the information below:
  0 No active event

  Bit 0 = 1 Part program start from channel status RESET

  Bit 1 = 1 Part program end

  Bit 2 = 1 Operator panel reset

  Bit 3 = 1 Ramp-up

  Bit 4 = 1 First start after the search run

  Bit 5-7 reserved, currently always 0

  With the general request to 0, it is possible to determine whether an event is present. If a running event disappears upon RESET, the associated display bit in the interface extinguishes. For very brief events, the corresponding bit remains for at least the duration of a complete PLC cycle.
- Each time MD20108 PROG_EVENT_MASK is reconfigured, /_N_CMA_DIR/_N_PROG_EVENT_SPF must be loaded or enabled. Otherwise, the alarm 14011 "Program _N_PROG_EVENT_SPF does not exist or not enabled for execution" is output.
- The display can be suppressed in the current block display using the DISPLOF attribute in the PROC statement.
- A single block stop can be disabled with SBLOF attribute command or via MD10702 IGNORE_SINGLEBLOCK_MASK (prevent single block stop) with Bit 0.

  The response to read-in disable and single-block processing can be controlled separately through the machine data MD20106 PROG_EVENT_IGN_SINGLEBLOCK (Prog events ignore the single block) and MD20107 PROG_EVENT_IGN_INHIBIT (Prog events ignore the read-in disable).

MD20106 PROG_EVENT_IGN_SINGLEBLOCK:
_N_PROG_EVENT_SPF causes a block change despite single block without a further start when

Bit 0 = 1 is set, after Part program start event

Bit 1 = 1 is set, after Part program end event

Bit 2 = 1 is set, after Operator panel reset event

Bit 3 = 1 is set, after Ramp-up event

Bit 4 = 1 is set, after First start after search run event

MD 20107: PROG_EVENT_IGN_INHIBIT:
_N_PROG_EVENT_SPF causes a block change despite read-in disable when

Bit 0 = 1 is set, after Part program start event

Bit 1 = 1 is set, after Part program end event

Bit 2 = 1 is set, after Operator panel reset event

Bit 3 = 1 is set, after Ramp-up event

Bit 4 = 1 is set, after First start after search run event

The following constraint applies for Bit 0 == 1 (program event after part program start):
If the program event ends with the part program command "RET", then RET always leads to an executable block (analogous to M17).

There is no new behavior for Bit 0 == 0, i.e. RET is interpreted in the interpreter and does not lead to an "executable block".

No sequences for **start/end of part program** are passed:

- If a user ASUP is started from the reset status, the described sequences for the event for start/end of part program are not passed.
- **Settable Prog-Event properties**

  Machine data MD20109 PROG_EVENT_MASK_PROPERTIES can be used to define further properties of "event-driven program calls" for specific channels:

  – Bit0 = 0: An ASUP started from the RESET channel state is followed by an "event-driven program call" as in earlier versions

  – Bit0 = 1: An ASUP started from the RESET channel state is not followed by an "event-driven program call"

With the **Part program start:**
/_N_CMA_DIR/_N_PROG_EVENT_SPF is executed as a subroutine.
_N_PROG_EVENT_SPF must be ended with M17 or RET. A return by means of REPOS command is not permitted and triggers alarm 16020 "Repositioning not possible".

Error **with operator panel reset** or **after ramp-up**:
If EMERGENCY STOP or an operating mode / NCK error is still present when the operator panel is reset or after rampup, then _N_PROG_EVENT_SPF will only be processed after EMERGENCY STOP has been acknowledged or the error has been acknowledged in the channel.

Assignment example

MD20106 PROG_EVENT_IGN_SINGLEBLOCK = 'H1F'

MD20107 PROG_EVENT_IGN_INHIBIT = 'HC'

MD20109 PROG_EVENT_MASK_PROPERTIES = 'H1'

Event programs

Example for call by all events

MD20108 PROG_EVENT_MASK = 'H0F' (event-driven program call),
i.e. call of _N_PROG_EVENT_SPF during part program start, part program end, operator panel reset and ramp-up:

PROC PROG_EVENT DISPLOF

Sequence for **part program start**

```
IF ($P_PROG_EVENT == 1)
N 10 R100 = 0                ;   Transfer parameters for machining cycles
N 20 M17
ENDIF
```

Sequence for **part program end** and **operator panel reset**

```
IF ($P_PROG_EVENT == 2) OR ($P_PROG_EVENT == 3)
N10 R20 = 5
N20 ENDIF
N30 M17
ENDIF
```

Sequence for **powerup**

```
IF ($P_PROG_EVENT == 4)
N10 $SA_SPIND_S[Ax4] = 0        ;   Speed for spindle start through virtual
                                    interface
N20 ENDIF
N30 M17
ENDIF
M17
```

Start with RESET key

One of the following part programs is automatically started with the RESET key:

- Whose name is in MD11620 PROG_EVENT_NAME (program name for Prog Event) and that has been stored in one of the /_N_CUS_DIR/ or /_N_CMA_DIR/ directories
- _N_PROG_EVENT_SPF (default).

Control via MD20107 PROG_EVENT_IGN_INHIBIT

If the following machine data settings are present:
MD20107 PROG_EVENT_IGN_INHIBIT= 'H04F'
MD20108 PROG_EVENT_MASK= 'H04F'

The program started with the RESET key is executed right up to the end independently of a possibly set read-in disable.

Note

Recommendation for MD11450 with block search:

MD11450 SEARCH_RUN_MODE = ’H7’ (search parameterization)

Bit 0 = 1:
With the loading of the last action block after block search, the processing is stopped and the VDI signal "Last action block active" is set. Alarm 10208 is not output until the PLC requests this by setting the VDI signal "PLC action ended".
Application: PLC starts an ASUP after block search.

Bit 1 = 1:
Automatic ASUP start after output of the action blocks (see also MD11620 PROG_EVENT_NAME). Alarm 10208 is not output until the ASUP is completed.

Bit 2 = 1:
Output of the auxiliary functions is suppressed in the action blocks. The spindle programming that accumulated during the block search can be output at a later point in time (e.g. in an ASUP).

The program data for this is stored in the following system variables:

- $P_SEARCH_S
- $P_SEARCH_SDIR
- $P_SEARCH_SGEAR
- $P_SEARCH_SPOS
- $P_SEARCH_SPOSMODE

8.3.9 Asynchronous subroutines (ASUPs)

Function

It is possible to activate two different ASUPs (PLCASUP1_SPF and PLCASUP2_SPF) from the PLC via the ASUP interface area. Before an asynchronous subroutine (ASUP) can be started from the PLC, it must have been assigned to an interrupt number by an NC program or by the PI service ASUP (see DB1200.DBB4000).

Once prepared in this way, it can be started at any time from the PLC. The NC program running is interrupted by the ASUP.

Only one ASUP can be started at one time. If the start signal for both ASUPs is to be set to logical 1 in a PLC cycle, the ASUPs are started in the sequence INT1 and then INT2.

The start signal must be set to logical 0 by the user once the ASUP has been completed or if an error has occurred.

Note

The call of the ASUP PI service must have been completed before an ASUP may be started.

Initialization

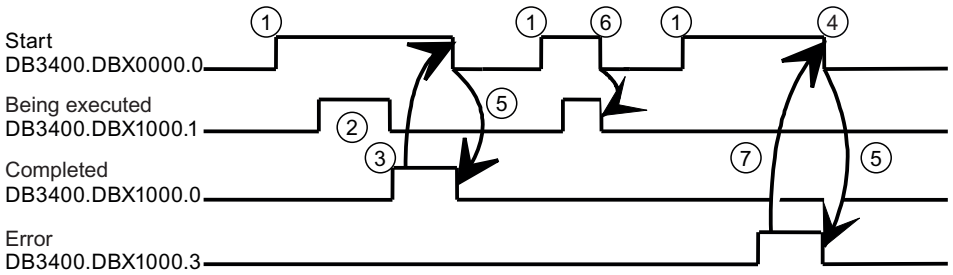
The initialization is performed via the ASUP PI service.

Starting an ASUP

The time sequence of an ASUP is shown in the following pulse diagram in the example of PLCASUP1.SPF. You can see from the table which interface signals are of relevance for PLCASUP2.SPF.

Table 8- 11 Assignment of the signals to the pulse diagram

Signal	Address - PLCASUP1_SPF	Address - PLCASUP2_SPF
Start	DB3400.DBX0000.0	DB3400.DBX0001.0
Being executed	DB3400.DBX1000.1	DB3400.DBX1001.1
Completed	DB3400.DBX1000.0	DB3400.DBX1001.0
Error	DB3400.DBX1000.3	DB3400.DBX1001.3
Interrupt no. not allocated	DB3400.DBX1000.2	DB3400.DBX1001.2



① Function activation via positive edge of Start
② ASUP is being executed
③ Positive acknowledgment: ASUP ended
④ Reset function activation after receipt of acknowledgment
⑤ Signal change through PLC
⑥ not permitted. If function activation is reset prior to receipt of acknowledgment, the output signals are not updated without the operational sequence of the activated function being affected
⑦ Negative acknowledgment: Error has occurred

Figure 8-3 Pulse diagram for PLCASUP1_SPF

Configuration

The behavior of the ASUP can be influenced via the following standard machine data.

- MD11602 ASUP_START_MASK (ignore stop reasons for ASUP)

  The machine data specifies which stop reasons are to be ignored for an ASUP start.

  Recommended: MD11602 = 'H7'

- MD11604 ASUP_START_PRIO_LEVEL (priority, as of which MD11602 is effective)

  This machine data specifies the ASUP priority as of which machine data MD11602 ASUP_START_MASK is to be applied. MD11602 is applied from the level specified here up to the highest ASUP priority level 1.

  Recommended: MD11604 = 2

- MD20116 IGNORE_INHIBIT_ASUP (execute interrupt program in spite of read-in disable)

  In spite of set read-in disable, an assigned user ASUP is processed completely for the interrupt channel with the set bit.

  Bit 0 is assigned to interrupt channel 1 (PLCASUP1)

  Bit 1 is assigned to interrupt channel 2 (PLCASUP2)

  The machine data is effective only if MD11602 ASUP_START_MASK Bit2 = 0

- MD20117 IGNORE_SINGLEBLOCK_ASUP (execute interrupt program completely in spite of single block)

  In spite of selected SBL processing mode, an assigned user ASUP is processed completely for the interrupt channel with the set bit.

  Bit 0 is assigned to interrupt channel 1 (PLCASUP1)

  Bit 1 is assigned to interrupt channel 2 (PLCASUP2)

  The machine data is effective only if

  MD10702 IGNORE_SINGLE_BLOCK_MASK Bit1 = 0

8.3.10 Responses to operator or program actions

Responses

The following table shows the channel and program states that result after certain operator and program actions.

The left-hand side of the table shows the channel and program states and the mode groups from which the initial situation can be selected. Various operator/program actions are listed on the right-hand side of the table, the number of the situation after the action has been carried out is shown in brackets after each action.

Table 8- 12 Responses to operator or program actions

Situation	Channel status			Program status				Active mode			Operator or program action (Situation after the action)
	R	U	A	N	U	Switch gear protection	A	A	M	J	
1		x					x	x			RESET (4)
2		x					x		x		RESET (5)
3		x					x			x	RESET (6)
4	x			x				x			NC Start (13); mode change (5 or 6)
5	x			x					x		NC Start (14); mode change (4 or 6)
6	x			x						x	Direction key (15); mode change (4 or 5)
7		x		x					x		NC Start (14)
8		x		x						x	NC Start (15)
9		x			x			x			NC Start (13); mode change (10 or 11)

Situation	Channel status			Program status				Active mode			Operator or program action (Situation after the action)
	R	U	A	N	U	Switch gear protection	A	A	M	J	
10		x			x				x		NC Start (16); mode change (9 or 11)
11		x			x					x	Direction key (17); mode change (9 or 10)
12		x				x		x			NC Start (13); mode change (10 or 11)
13			x				x	x			NC Stop (12)
14			x	x					x		NC Stop (7); at block end (5)
15			x	x						x	NC Stop (8); at JOG end (6)
16			x		x				x		NC Stop (10); at block end (10)
17			x		x					x	NC Stop (11); at JOG end (11)

Description

Channel status:
R: aborted
U: interrupted
A: running

Program status:
N: aborted
U: interrupted
S: stopped
A: running

Operating modes:
A: AUTO
M: MDA
J: JOG

8.3.11 Example of a timing diagram for a program run

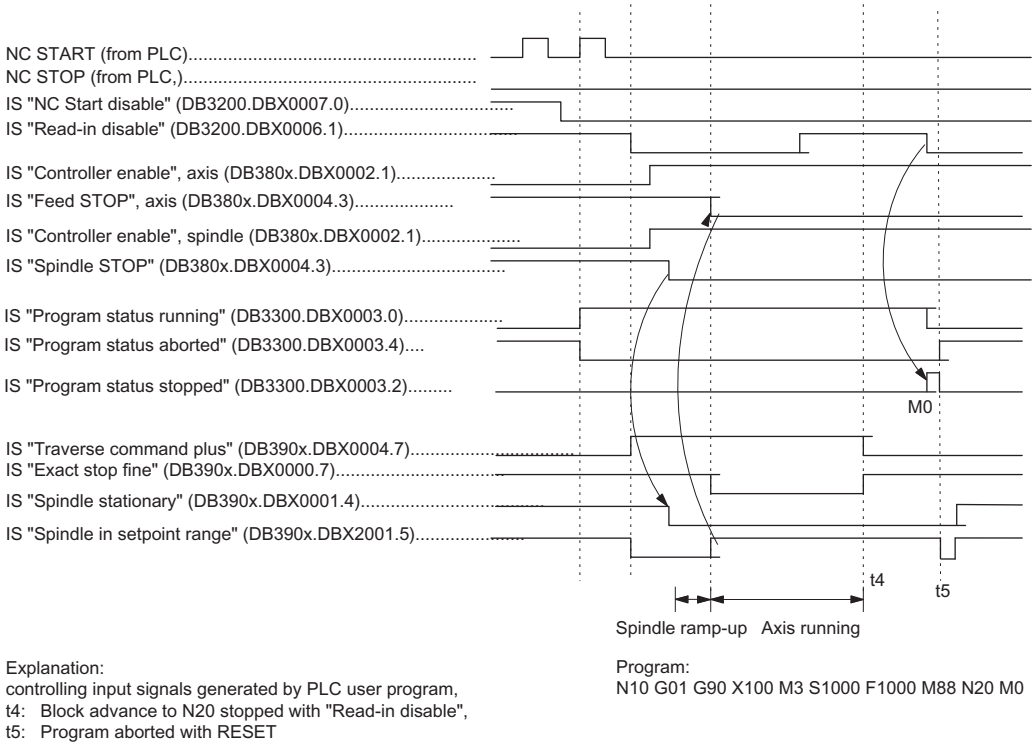


Figure 8-4 Examples of signals during a program run

8.4 Program test

8.4.1 General information on the program test

Purpose

Several control functions are available for testing a new part program. These functions are provided to reduce danger at the machine and time required for the test phase. It is possible to activate several program test functions simultaneously.

The following test options are described here:

- Program processing without axis movements
- Program processing in single-block mode
- Program processing with dry run feedrate
- Processing of certain program sections
- Skipping certain program parts
- Graphic simulation

8.4.2 Program processing without axis movements (PRT)

Functionality

The part program can be started and processed with active "Program test" function via the IS "NC Start" (DB3200.DBX0007.1), i.e. with auxiliary function outputs, dwell times. Only the axes/spindles are simulated. The software limit switch safety function continues to be valid.

The position control is not interrupted, so the axes do not have to be referenced when the function is switched off.

The user can check the programmed axis positions and auxiliary function outputs of a part program.

Note

Program processing without axis motion can also be activated with the function "Dry run feedrate".

Selection/activation

This function is selected via the user interface in the menu "Program control". IS "Program test selected" (DB1700.DBX0001.7) is set on selection of the function.

The PLC user program must activate the function via the IS "Activate program test" (DB3200.DBX0001.7).

Display

As a checkback for the active program test, "PRT" is displayed in the status line on the user interface and the IS "Program test active" (DB3300.DBX0001.7) is set in the PLC.

8.4.3 Program processing in single block mode (SBL)

Functionality

The user can execute a part program block-by-block to check the individual machining steps. Once the user decides that an executed part program block is functioning correctly, he/she can call the next block. The program is advanced to the next part program block via IS "NC Start" (DB3200.DBX0007.1).

When the function "single block" is activated, the part program stops after every program block during processing. In this case the activated single block type must be observed.

Single-block type

The following different types of single block are provided:

- Single block, coarse

  With this type of single block, the blocks that initiate actions (traversing motions, auxiliary function outputs, etc.) are processed individually. If tool radius compensation is active (G41,G42), processing stops after every intermediate block inserted by the control. Processing is however not stopped at calculation blocks as these do not trigger actions.

- Single block, fine

  With this type of single block, **all** blocks of the part program (even the pure computation blocks without traversing motions) are processed sequentially by NC Start.

"Single block coarse" is the default setting after switching on.

⚠ CAUTION
In a series of G33 blocks single block is effective only if "dry run feedrate" is selected.

Selection/activation

The selection signal normally comes from a user machine control panel.

This function must be activated by the PLC user program via the IS "Activate single block" (DB3200.DBX0000.4).

The preselection whether "Single block coarse" or "Single block fine" type is made in the user interface in the "Program control" menu.

Display

The checkback signal that single block mode is active is displayed in the relevant "SBL" field on the operator interface.

Because of the single block mode, as soon as the part program processing has processed a part program block:

- The following interface signals are set:
  - IS "Channel status interrupted" (DB3300.DBX0003.6)
  - IS "Program status stopped" (DB3300.DBX0003.2)
- The following interface signals are reset:
  - IS "Channel status active" (DB3300.DBX0003.5)
  - IS "Program status running" (DB3300.DBX0003.0)

8.4.4 Program processing with dry run feedrate (DRY)

Functionality

The part program can be started via IS "NC Start" (DB3200.DBX0007.1). When the function is active, the traversing velocities programmed in conjunction with G1, G2, G3, CIP, and CT are replaced by the feed value stored in SD42100 DRY_RUN_FEED. The dry run feedrate also replaces the programmed revolutional feedrate in program blocks with G95. However, if the programmed feedrate is larger than the dry run feedrate, then the larger value is used.

NOTICE
Damage to the workpiece or machine tool Workpieces may not be machined when "dry run feedrate" is active because the altered feedrates might cause the permissible tool cutting rates to be exceeded and the workpiece or machine tool could be damaged.

Selection/activation

Operation with dry run feedrate is selected in the "MACHINE" operating area -> "Program control" softkey ("AUTO" mode). IS "Dry run feedrate" (DB1700.DBX0000.7) is set on selection of the function. In addition, the required dry run feedrate must be entered in the menu "Set data". This does not activate the function.

This function is activated via the IS "Activate dry run feedrate" (DB3200.DBX0000.4) and is evaluated at NC start.

The dry run feedrate must be entered before program start in SD42100 DRY_RUN_FEED.

Display

The checkback signal that dry run feedrate is active is displayed in the relevant "DRY" status line on the user interface.

8.4.5 Block search: Processing of certain program sections

Functionality

To set the program run to a certain block (target block) of a part program, the block search function can be used. It can be selected whether or not the same calculations are to be performed during the block search up to the target block as would be performed during normal program operation.

After the target block is reached, the part program can be started via IS "NC Start" (give 2x) (DB3200.DBX0007.1). If necessary there is an automatic compensating movement of the axes to start or end positions of the target block. Execution of the remaining program then continues.

Note

Pay attention to a collision-free start position and appropriate active tools and other technological values! If necessary, a collisionfree start position must be approached manually with JOG. Select the target block considering the selected block search type.

Selection/activation

The block search is selected in the "AUTO" mode on the user interface.

The search run can be activated with corresponding softkey for the following functions:

- Block search with calculation to contour

  Is used in any circumstances in order to approach the contour. On NC Start, the **start position of the target block** or the end position of the block before the target block is approached. This is traversed up to the end position. Processing is true to contour.

- Block search with calculation to block end point

  Is used in any circumstances in order to approach a target position (e.g. tool change position). The **end position of the target block** or the next programmed position is approached using the type of interpolation valid in the target block. This is not true to contour. Only the axes programmed in the target block are moved.

- Block search without calculation.

  Is used for a quick search in the main program. No calculations are performed. The internal controller values indicate the status valid before the search. Whether the program can be executed subsequently depends on the program and must be decided by the operator. This search run is suitable for a fast syntax check of a new program.

Interface signal

In the PLC, the following interface signals are set according to a time sequence (see figure):

- "Block search active" (DB3300.DBX0001.4)
- "Action block active" (DB3300.DBX0001.3)
- "Approach block active" (DB3300.DBX0000.4)

Note

The "Approach block active" is only enabled with "Block search with calculation on contour" because a separate approach block is not generated with "Block search with calculation at block end point" (the approach block is the same as the target block).

- "Last action block active" (DB3300.DBX0000.6)

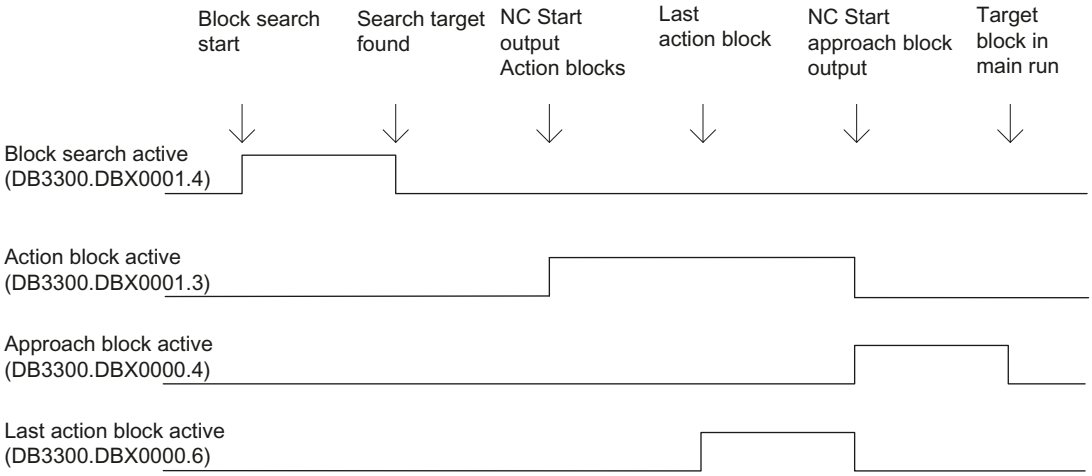


Figure 8-5 Chronological order of interface signals

After "Block search with calculation at block end point", automatic repositioning is not performed between "Last action block active" and continuation of part program processing by NC Start. The start point of the approach movement is the current axis position on NC Start; the end point results from the processing of the part program.

Action blocks

Action blocks contain the actions accumulated during "block search with calculation", e.g. auxiliary function outputs, and tool (T, D), spindle (S) and feed programming commands. During "block search with calculation" (contour or block end point), actions such as M function outputs are accumulated in so-called "action blocks". These blocks are output on an NC Start after "Search target found".

Note

The action blocks also activate the accumulated spindle programming (S value, M3/M4/M5, SPOS). The PLC user program must ensure that the tool can be operated and, if necessary, the spindle programming is reset via the IS "Spindle reset" (DB380x.DBX0002.2).

PLC actions after block search

There is the IS "Last action block active" to enable activation of PLC actions after block search. The signal indicates that all action blocks have been executed and it is now possible to perform PLC actions or operator actions (e.g. mode change). This allows the PLC to perform another tool change, for example, before the start of the movement.

The alarm 10208 is also output per default at this time. It should indicate to the operator that an NC start is still necessary to continue program processing.

Supplementary condition

The approach movement "Search with calculation to block end point" is performed using the type of interpolation valid in the target block. This should be G0 or G1, as appropriate. With other types of interpolation, the approach movement can be aborted with an alarm (e.g. circle end point error on G2/G3).

Note

For further information about the block search function, refer to the SINUMERIK 808D Programming and Operating Manual.

8.4.6 Skip part program blocks (SKP)

Functionality

When testing or breaking in new programs, it is useful to be able to disable or skip certain part program blocks during program execution.

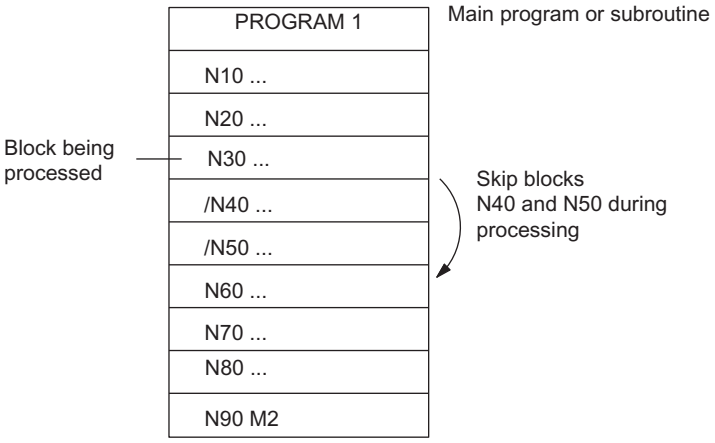


Figure 8-6 Skipping part program blocks

Selection/activation

The skip function is selected through the user interface in the menu "Program control". IS "Skip block selected" (DB1700.DBX0002.0) is set when the function is selected. In addition, a slash "/" must be written before the blocks to be skipped (see figure). This however does not activate the function.

This function is activated via IS "Activate skip block" (DB3200.DBX0002.0).

Display

The checkback signal that the "Skip block" function is active is displayed in the relevant "SKP" status line on the user interface.

8.4.7 Graphic simulation

Function

In the "AUTO" operating mode a selected and opened program can be simulated graphically on the screen of the control unit. The movements of the programmed axes are recorded as line diagram after an NC start.

Selection/deselection

The graphic simulation can be reached for the selected program through the "PROGRAM" operating area, open program and "Simulation" softkey. Here the IS "Simulation active" (DB1900.DBX0000.6) is set and reset again on leaving the "PROGRAM" operating area or changing to "Edit".

Display

Due to numerous operating possibilities a complete workpiece, or else only enlarged details of it, can be displayed on the screen.

Reference:

SINUMERIK 808D Programming and Operating Manual

PLC user program

The PLC user program must itself influence the required behavior of the control system in simulation, for example:

- Stop axes/spindle by transition into the program test: Set IS "Activate program test" (DB3200.DBX0001.7)
- Abort the running program if "Simulation" is exited by setting IS "Reset" (DB3000.DBX0000.7), etc.

Display machine data

A number of display machine data (MD283 to MD292) is available for the user-specific configuration of the graphic simulation.

Reference:

SINUMERIK 808D Parameter Manual

8.5 Timers for program execution time

Function

Timers are provided under the "Program execution time" function and these can be used for monitoring technological processes in the program or only in the display. These timers are read-only.

There are timers that are always active. Others can be deactivated via machine data.

Timers - always active

- Time since the last "Control powerup with default values" (in minutes):

  $AN_SETUP_TIME

  The timer is automatically reset to zero in the case of a "Control power-up with default values".

- Time since the last control powerup (in minutes):

  $AN_POWERON_TIME

  It is reset to zero automatically with each power-up of the control system.

Timers that can be deactivated

The following timers are activated via the machine data (default setting). The start is timer-specific. Each active run-time measurement is automatically interrupted in the stopped program state or for feedrate-override = zero.

The behavior of the activated timers for active dry run feedrate and program testing can be specified using machine data.

- Total execution time in seconds of NC programs in the "AUTO" mode (in seconds):

  $AC_OPERATING_TIME

  In the "AUTO" mode, the runtimes of all programs between NC start and end of program / RESET are summed up. The timer is zeroed with each power-up of the control system.

- Runtime of the selected NC program (in seconds):

  $AC_CYCLE_TIME

  The runtime between NC Start and End of program / Reset is measured in the selected NC program. The timer is reset with the start of a new NC program.

- Tool action time (in seconds):

  $AC_CUTTING_TIME

  The runtime of the path axes is measured in all NC programs between NC START and end of program / RESET without rapid traverse active and with the tool active. The measurement is interrupted when a dwell time is active. The timer is automatically reset to zero in the case of a "Control power-up with default values".

Display

The contents of the timers are visible on the screen in the "OFFSET" operating area -> "Set data" softkey -> "Time counter" softkey:

- **Run time** = $AC_OPERATING_TIME
- **Cycle time** = $AC_CYCLE_TIME
- **Cutting time** = $AC_CUTTING_TIME
- **Setup time** = $AN_SETUP_TIME
- **Power on time** = $AN_POWERON_TIME

"Cycle time" is also visible in the information line of the "AUTO" window of the "MACHINE" operating area.

Reference:

Programming and Operating Manual

8.6 Workpiece counter

Function

The "Workpiece counter" function provides counters for counting workpieces. These counters can be read and written by the program or by operation (note protection level for writing).

Range of values: 0 to 999 999 999.

The following channel-specific machine data can be used to control counter activation, counter reset timing and the counting algorithm.

- MD27880 PART_COUNTER (activation of workpiece counters)
- MD27882 PART_COUNTER_MCODE (workpiece counting with user-defined M command)

Counter

- Number of workpieces required (workpiece target):

  $AC_REQUIRED_PARTS

  In this counter you can define the number of workpieces at which the actual workpiece counter $AC_ACTUAL_PARTS is reset to zero.

  MD27880 PART_COUNTER (Bit 0) can be used to generate the display alarm 21800 "Required number of workpieces reached" and to output the IS "Required number of workpieces reached" (DB3300.DBX40001.1).

- Total number of workpieces produced (total actual):

  $AC_TOTAL_PARTS

  The counter specifies the total number of all workpieces produced since the start time.

- Number of actual workpieces (current actual):

  $AC_ACTUAL_PARTS

  This counter registers the number of all workpieces produced since the starting time. The counter is automatically reset to zero (on condition that $AC_REQUIRED_PARTS is not equal to 0) when the required number of workpieces ($AC_REQUIRED_PARTS) has been reached.

- Number of workpieces specified by the user:

  $AC_SPECIAL_PARTS

  This counter allows users to make a workpiece counting in accordance with their own definition. Alarm output can be defined for the case of identity with $AC_REQUIRED_PARTS (workpiece target). Users must reset the counter themselves.

The first output of the M command for counting after resetting the counter applies as start point. This M command is set in MD27880 PART_COUNTER or MD27882 PART_COUNTER_MCODE for the relevant counter.

Display

The contents of the counters are visible on the screen in the "OFFSET" operating area -> "Set data" softkey -> "Timer counter":

- **Part total** = $AC_TOTAL_PARTS
- **Part required** = $AC_REQUIRED_PARTS
- **Part count** = $AC_ACTUAL_PARTS

  ($AC_SPECIAL_PARTS not available for display)

"Part count" is also visible in the information line of the "Auto" window of the "MACHINE" operating area.

References:

Programming and Operating Manual

8.7 Data table

8.7.1 Machine data

NC-specific machine data

Number	Identifier	Name
General		
10702	IGNORE_SINGLEBLOCK_MASK	Prevent single-block stop
11450	SEARCH_RUN_MODE	Block search parameter settings
11602	ASUP_START_MASK	Ignore stop conditions for ASUP
11604	ASUP_START_PRIO_LEVEL	Priorities for ASUP_START_MASK
11620	PROG_EVENT_NAME	Program name for a program event

Basic machine data of the channel

Number	Identifier	Name
Channel-specific		
20050	AXCONF_GEOAX_ASSIGN_TAB[n]	Assignment between geometry axis and channel axis [GEOaxis no.]: 0...2
20060	AXCONF_GEOAX_NAME_TAB[n]	Geometry axis name in channel [GEOaxis no.]: 0...2
20070	AXCONF_MACHAX_USED[n]	Machine axis number valid in channel [channel axis no.]: 0...4
20080	AXCONF_CHANAX_NAME_TAB[n]	Channel axis name in channel [channel axis no.]: 0...4

Number	Identifier	Name
20100	DIAMETER_AX_DEF	Geometry axis with transverse axis function
20106	PROG_EVENT_IGN_SINGLEBLOCK	Prog events ignore the single block
20107	PROG_EVENT_IGN_INHIBIT	Prog events ignore the read-in disable
20108	PROG_EVENT_MASK	Eventdriven program calls
20109	PROG_EVENT_MASK_PROPERTIES	Prog event properties
20110	RESET_MODE_MASK	Initial setting at RESET
20112	START_MODE_MASK	Initial setting at special NC Start after power-up and at RESET
20116	IGNORE_INHIBIT_ASUP	Execute user ASUPs completely in spite of readin disable
20117	IGNORE_SINGLEBLOCK_ASUP	Process user ASUPs completely in spite of single-block processing
20700	REFP_NC_START_LOCK	NC-Start disable without reference point
21000	CIRCLE_ERROR_CONST	Circle end point monitoring constant
20150	GCODE_RESET_VALUES	Reset G groups
20152	GCODE_RESET_MODE	G code basic setting at RESET

Auxiliary function settings of the channel

Number	Identifier	Name
Channel-specific		
22000	AUXFU_ASSIGN_GROUP[n]	Auxiliary function group [aux. func. no. in channel]: 0...63
22010	AUXFU_ASSIGN_TYPE[n]	Auxiliary function type [aux. func. no. in channel]: 0...63
22020	AUXFU_ASSIGN_EXTENSION[n]	Auxiliary function extension [aux. func. no. in channel]: 0...63
22030	AUXFU_ASSIGN_VALUE[n]	Auxiliary function value [aux. func. no. in channel]: 0...63
22550	TOOL_CHANGE_MODE	New tool offset for M function

Timers and counters of the channel

Number	Identifier	Name
Channel-specific		
27860	PROCESSTIMER_MODE	Activation of the program runtime measurement
27880	PART_COUNTER	Activation of the workpiece counters
27882	PART_COUNTER_MCODE[n]	Workpiece counting via M command, n = 0 ... 2

Display machine data

Number	Identifier	Name
283 ... 292		Setting of the display for the graphic simulation

8.7.2 Setting data

Number	Identifier	Name
Channel-specific		
42000	THREAD_START_ANGLE	Start angle for thread
42010	THREAD_RAMP_DISP	Starting and deceleration distance of feed axis in thread cutting G33
42100	DRY_RUN_FEED	Dry run feedrate

8.7.3 Interface signals

Operating mode signals

Number	Bit	Name
PLC to NCK		
DB3000.DBX0000	.0	AUTO mode
DB3000.DBX0000	.1	MDA mode
DB3000.DBX0000	.2	JOG mode
DB3000.DBX0000	.4	Mode change disable
DB3000.DBX0000	.7	RESET
DB3000.DBX0001	.2	Machine function REF
NCK to PLC		
DB3100.DBX0000	.0	Active mode AUTO
DB3100.DBX0000	.1	Active mode MDA
DB3100.DBX0000	.2	Active JOG mode
DB3100.DBX0000	.3	808D READY
DB3100.DBX0001	.2	Active machine function REF

Channel signals

Number	Bit	Name
PLC to NCK		
DB3200.DBX0000	.4	Activate single block
DB3200.DBX0000	.5	Activate M01
DB3200.DBX0000	.6	Activate dry run feed
DB3200.DBX0001	.0	Activate referencing
DB3200.DBX0001	.7	Activate program test
DB3200.DBX0002	.0	Block skip
DB3200.DBX0006	.0	Feed disable
DB3200.DBX0006	.1	Read-in disable
DB3200.DBX0006	.2	Delete distance-to-go
DB3200.DBX0006	.3	Delete UP number of passes
DB3200.DBX0006	.4	Program level abort
DB3200.DBX0006	.6	Rapid traverse override active
DB3200.DBX0006	.7	Feed rate override active
DB3200.DBX0007	.0	NC Start disable
DB3200.DBX0007	.1	NC Start
DB3200.DBX0007	.2	NC Stop at block limit
DB3200.DBX0007	.3	NC stop
DB3200.DBX0007	.4	NC Stop axes plus spindles
DB3200.DBX0007	.7	Reset
NCK to PLC		
DB3300.DBX0000	.3	Action block active
DB3300.DBX0000	.4	Approach block active
DB3300.DBX0000	.5	M00/M01 active
DB3300.DBX0000	.6	Last action block active
DB3300.DBX0001	.0	Referencing active
DB3300.DBX0001	.4	Block search active
DB3300.DBX0001	.5	M2 / M30 active
DB3300.DBX0001	.7	Program test active
DB3300.DBX0003	.0	Program status: Running
DB3300.DBX0003	.2	Program status: Stopped
DB3300.DBX0003	.3	Program status: Interrupted
DB3300.DBX0003	.4	Program status: Aborted
DB3300.DBX0003	.5	Channel status: Active
DB3300.DBX0003	.6	Channel status: Interrupted
DB3300.DBX0003	.7	Channel status: Reset
DB3300.DBX4001	.1	Workpiece target reached
HMI to PLC		
DB1700.DBX0000	.5	M01 selected
DB1700.DBX0000	.6	Dry run feed rate selected

Number	Bit	Name
DB1700.DBX0001	.3	Feed rate override selected for rapid traverse
DB1700.DBX0001	.7	Program test selected
DB1700.DBX0002	.0	Skip-block selected
DB1900.DBX0000	.6	Simulation active

ASUP signals

Number	Bit	Name	
PLC to NCK			
DB3400.DBX0000	.0	INT1 Start	
DB3400.DBX0001	.0	INT2 Start	
DB3400.DBX1000	.0	ASUP ended	INT1
DB3400.DBX1000	.1	ASUP is being executed	
DB3400.DBX1000	.2	Interrupt no. not allocated	
DB3400.DBX1000	.3	ASUP version not possible	
DB3400.DBX1001	.0	ASUP ended	INT2
DB3400.DBX1001	.1	ASUP is being executed	
DB3400.DBX1001	.2	Interrupt no. not allocated	
DB3400.DBX1001	.3	ASUP version not possible	

Compensation 9

9.1 Brief description

Compensations

For SINUMERIK 808D, the following axis-specific compensation functions can be activated:

- Backlash compensation
- Interpolatory compensation
  - leadscrew error and measuring system error compensation (LEC)

These compensation functions can be set for each machine individually with axis-specific machine data.

Position display

The normal actual-value and setpoint position displays ignore the compensation values and show the position values of an ideal machine. To view the compensation values, enter the "SYSTEM" operating area -> "Serv. displ." -> "Service axes", and navigate to the item "Abs. compens. value meas. system 1".

9.2 Backlash compensation

Effect

In the case of axes/spindle with indirect measuring systems, mechanical backlash results in corruption of the traverse path, causing an axis, for example, to travel too much or too little by the amount of the backlash when the direction of movement is reversed (see following figure).

Compensation

To compensate for backlash, the axis-specific actual value is corrected by the amount of backlash every time the axis/spindle changes direction.

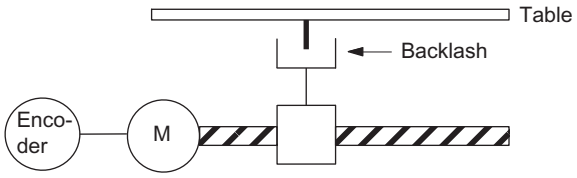
This quantity can be entered for each axis/spindle at the commissioning phase in MD32450 BACKLASH (backlash compensation)

Effectiveness

Backlash compensation is always active in all operating modes after reference point approach.

Positive backlash

The encoder leads the machine part (e.g. table). Since the actual position acquired by the encoder also leads the real actual position of the table, the table travels too short a distance (see diagram below). The backlash compensation value must be entered as a **positive** value here (= normal case).



Encoder actual value leads the real actual value (table):
The table does not traverse far enough

Figure 9-1 Positive backlash (normal case)

Negative backlash

The encoder lags behind the machine part (e.g. table); the table then travels too far. The correction value entered is **negative**.

High backlash compensation values

The user has the option of applying the backlash compensation value gradually in several increments when the relevant axis reverses direction. This prevents an excessive setpoint step change from causing specific axis errors.

The contents of the axis-specific MD36500 ENC_CHANGE_TOL determine the increment with which the backlash compensation value (MD32450 BACKLASH) is applied. Please note that the backlash compensation is fully calculated only after n servo cycles (n=MD32450/MD36500). An excessive time span can cause the triggering of standstill monitoring alarms. If MD36500 is greater than MD32450, the compensation is performed in a servo cycle.

9.3 Interpolatory compensation

9.3.1 General

Terminology

Compensation value: The difference between the axis position measured by the position actual-value encoder and the required programmed axis position (= axis position of the ideal machine). The compensation value is often also referred to as the correction value.

Interpolation point: A position of the axis and the corresponding offset value.

Offset table: Table containing interpolation points

Compensation table

Because dimensional deviations between the leadscrew pitch and the measuring system directly affect the accuracy of workpiece machining, they must be compensated for by the relevant position-dependent compensation values. The compensation values are derived from measured error curves and entered in the control in the form of compensation tables during installation. A separate table must be created for each compensation relation.

The compensation values and additional table parameters are entered in the compensation tables using special system variables.

Entry of compensation table

Compensation tables can be loaded to the backed up NC user memory by two different methods.

- The compensation values are loaded when an NC program tables is started. with the compensation
- The compensation values can also be loaded by transferring the tables from a personal computer (PC) through the serial interface on the HMI.

Note

The compensation tables can be output via the serial interface on the HMI from operating area "SYSTEM" -> "Sys. data" / NCK/PLC data: Leadscrew error compensation and loaded back following editing.

Linear interpolation between interpolation points

The traversing path to be compensated - defined using the start and end positions - is divided up into several (number depends on error curve shape) path segments of equal size (see figure below). The actual positions that limit these sub-paths are designated "interpolation points". A compensation value must be entered for each interpolation point (actual position) during commissioning. The compensation value applied between two interpolation points is generated on the basis of **linear interpolation** using the compensation values for the adjacent interpolation points (i.e. adjacent interpolation points are linked along a line).

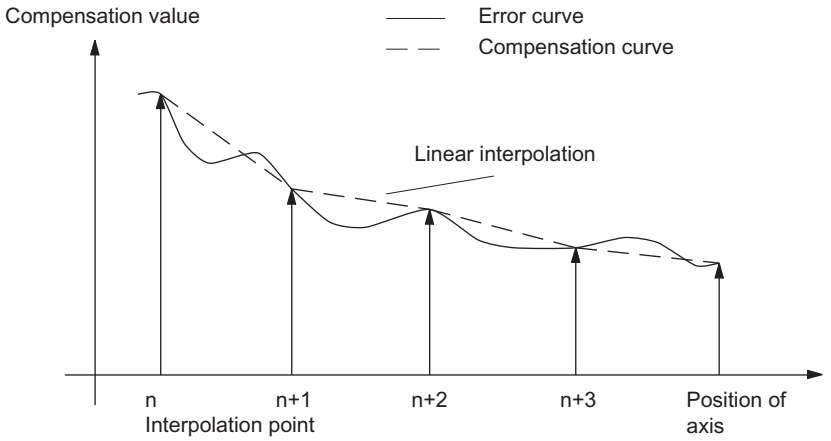


Figure 9-2 Linear interpolation between the interpolation points

Compensation value at reference point

The compensation table should be structured such that the compensation value at the reference point is "zero". This prevents position jumps occurring when the LEC is activated (after reference point approach).

9.3.2 LEC

Function

The leadscrew error compensation / measuring system error compensation (LEC) is an axis-specific compensation.

The principle of the LEC is to modify the axis-specific position actual value by the assigned compensation value in the interpolation cycle and to apply this value to the machine axis for immediate traversal. A positive compensation value causes the corresponding machine axis to move in the negative direction.

The magnitude of the compensation value is not limited and is not monitored. In order to avoid impermissibly high velocities and accelerations caused by compensation, small compensation values must be selected. Large compensation values can cause other axis monitoring functions to output alarms (e.g. contour monitoring, velocity setpoint limitation).

Effectiveness

- The compensation values are stored in the NC user memory and active (after POWER ON).
- The function has been activated for the relevant machine axis

  (MD32700 ENC_COMP_ENABLE [0] = 1).
- The axis has been referenced (IS "Referenced/synchronized 1" DB390x.DBX0000.4 set).

As soon as these conditions have been fulfilled, the axis-specific actual value is altered by the compensation value in all modes and traversed by the machine axis immediately.

If the reference is then lost, e.g. because the encoder frequency has been exceeded (IS "Referenced/synchronized 1" =0), compensation processing is de-activated.

Compensation table

The position-related compensation values are stored in the form of system variables for the relevant axis in the compensation table. 125 interpolation points (N = 0...124) are possible.

The following measuring-system-specific parameters must be set for the table (see Fig. "Compensation table parameters (system variables for LEC)"):

- **Compensation value for interpolation point N in compensation table:**

  $AA_ENC_COMP [0,N,AXi]= ...

  where: AXi = machine axis name, e.g. X1, Y1, Z1; N = interpolation point index

  For every individual interpolation point (axis position) the compensation value must be entered in the table. The magnitude of the compensation value is not limited.

  Note

  The first and last compensation values remain active over the entire traversing range; i.e. these values should be set to "0" if the compensation table does not cover the entire traversing range.

- **Distance between interpolation points:** $AA_ENC_COMP_STEP[0,AXi]= ...

  The distance between interpolation points defines the distance between the compensation values in the relevant compensation table (see above for AXi).

- **Starting position:** $AA_ENC_COMP_MIN[0,AXi]= ...

  The starting position is the axis position at which the compensation table for the relevant axis begins (interpolation point 0).

  The compensation value for the starting position is $AA_ENC_COMP[0,**0**,AXi].

  The compensation value of interpolation point 0 is used for all positions smaller than the starting position (exception: table with modulo function).

- **End position:** $AA_ENC_COMP_MAX[0,AXi]= ...

  The end position is the axis position at which the compensation table for the relevant axis ends (interpolation point k < 125).

  The compensation value for the end position is $AA_ENC_COMP[0,**k**,AXi]

  The compensation value of interpolation point k is used for all positions larger than the end position (exception: table with modulo function). Compensation values which are greater than k are inactive.

- **Compensation with modulo function**: $AA_ENC_COMP_IS_MODULO[0,AXi] = 1

  When compensation with modulo function is activated, the compensation table is repeated cyclically; i.e. the compensation value at position $AA_ENC_COMP_MAX (interpolation point $AA_ENC_COMP[0,**k**,AXi]) is immediately followed by the compensation value at position $AA_ENC_COMP_MIN (interpolation point $AA_ENC_COMP[0,**0**,AXi]).

  For rotary axes with modulo 360° it is therefore suitable to program 0° ($AA_ENC_COMP_MIN) as the starting position and 360° ($AA_ENC_COMP_MAX) as the end position. In this case both compensation values must be entered directly.

⚠ CAUTION

When the compensation values are entered, it is important that all interpolation points within the defined range be assigned a compensation value (i.e. there should be no gaps). Otherwise, the compensation value that was left over from previous entries at these positions is used for these interpolation points.

Note

Table parameters that contain position data are interpreted through MD10240 SCALING_SYSTEM_IS_METRIC=0 in inches.

The position data can be automatically re-calculated by performing a manual switchover.

The compensation table can only be loaded when MD32700 ENC_COMP_ENABLE=0 has been set. The value "1" causes the compensation to be activated and write protection to be applied (output alarm 17070).

Example

The following example shows compensation value inputs for machine axis X1 as a program.

```
%_N_AX_EEC_INI
CHANDATA(1)
$AA_ENC_COMP[0,0,X1]=0.0            ; 1st compensation value (interpolation point
                                    0) +0 mm
$AA_ENC_COMP[0,1,X1] = 0.01         ; 2nd compensation value (interpolation point
                                    1) +10 mm
$AA_ENC_COMP[0,2,X1]=0.012          ; 3rd compensation value (interpolation point
                                    2) +12 mm
...
$AA_ENC_COMP[0,120,X1]=0.0          ; last compensation value (interpolation point
                                    120)

$AA_ENC_COMP_STEP[0,X1]=2.0         ; distance between interpolation points 2.0 mm
$AA_ENC_COMP_MIN[0,X1]=-200.0       ; compensation starts at -200.0 mm
$AA_ENC_COMP_MAX[0,X1] = 40.0       ; compensation ends at +40.0 mm
$AA_ENC_COMP_IS_MODULO[0,X1]=0      ; compensation without modulo function M17
```

Values for more than 125 interpolation points result in alarm 12400 "Element does not exist".

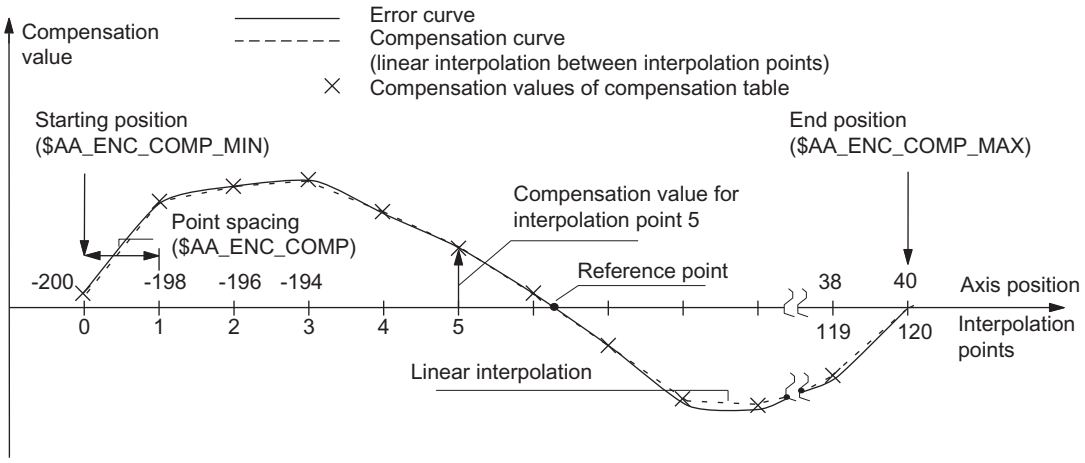


Figure 9-3 Compensation table parameters (system variables for LEC)

9.4 Data table

9.4.1 Machine data

Number	Identifier	Name
Axis-specific		
32450	BACKLASH[0]	Backlash
32630	ACTIVATION_MODE	Feedforward control can be activated from the program
32700	ENC_COMP_ENABLE[0]	Interpolatory compensation active
32810	EQUIV_SPEEDCTRL_TIME[0]...[5]	Equivalent time constant of the speed control loop
36500	ENC_CHANGE_TOL	Backlash compensation partial section
38000	MM_ENC_COMP_MAX_POINTS[0]	Interpolation points for encoder/spindle compensation (LEC) (for display only)

9.4.2 Interface signals

Number	Bit	Name
Axis/spindle-specific		
DB390x.DBX0000	.4	Referenced/synchronized 1

Measurement 10

10.1 Brief description

Channel-specific measuring

A measurement mode is programmed in a part program block (with or without DDTG). A trigger event (edge of the probe) is defined additionally, which will trigger the measurement process. The instructions apply to all axes programmed in this particular block. The program with the measurement process in "AUTO" mode is executed and can be employed for workpiece or tool measuring.

Tool measuring in JOG

SINUMERIK 808D includes operator support for the measurement process in "JOG" mode specially for measuring tools. Channel-specific measuring is integrated into this sequence. The PLC user program must include the required functionality. The measured offset values of the tool are available in the tool offset memory at the end of the measuring sequence.

The exact operating instructions are contained in the SINUMERIK 808D Programming and Operating Manual.

Note

The automatic measuring function is supported only on a milling machine.

10.2 Hardware requirements

10.2.1 Probes that can be used

General

In order to measure tool and workpiece dimensions, a touch-trigger probe is required that supplies a constant signal (rather than a pulse) when deflected.

The probe must operate virtually bounce-free. Most sensors can be adjusted mechanically to ensure that they operate in this manner.

Different types of probes supplied by a variety of manufacturers are available on the market. Probes are therefore divided into three groups according to the number of directions in which they can be deflected (see figure below).

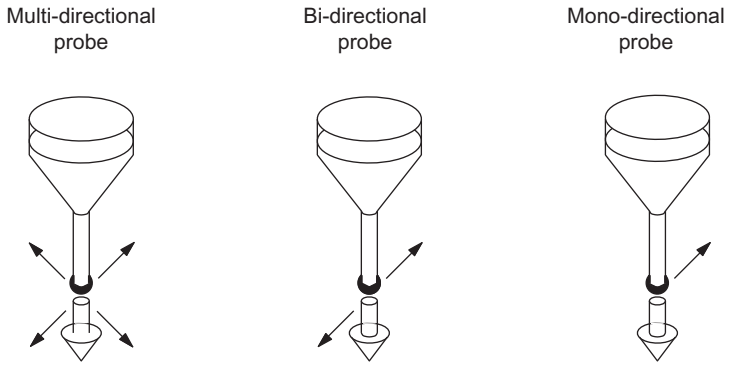


Figure 10-1 Probe types

Table 10- 1 Probe assignment

Probe type	Milling and machining centers
	Workpiece measurements
Multi-directional	X
Bi-directional	X
Mono-directional	X

A mono-probe can also be used for this purpose for milling and machining centers.

Multidirectional probe (3D)

This probe type can be used unconditionally for measuring tool and workpiece dimensions.

Bidirectional probe

This probe type is handled in the same way as a mono probe in milling and machining centers.

Mono-directional probe

This probe type can be used, with only a few restrictions, to take workpiece measurements on milling and machining centers.

The spindle must be capable of being positioned with the **SPOS** NC function if the measurement is to be carried out in different axis directions/axes. The probe must therefore be aligned according to the measuring task.

Switching performance

The signal level of the connected probe (deflected/non-deflected condition) must be communicated to the control via the MD13200 MEAS_PROBE_LOW_ACTIVE[0].

10.2.2 Probe connection

The probe for the SINUMERIK 808D is connected to pin4 (DI1) and pin5 (DI2) of X21. The actually used pin is determined by the relevant macro command. Thus, all measuring inputs of the axis drive modules are operated whose axes are involved in measuring. For the probe, use an external voltage (24 V) whose reference potential should be connected to X21, pin 10.

To ensure optimum interference immunity when connecting probes, lines must be used.

Reference:

SINUMERIK 808D Electrical Installation Manual

10.3 Channel-specific measuring

10.3.1 Measuring mode

Measuring commands MEAS and MEAW

The measuring operation is activated from the part program. A trigger event and a measuring mode are programmed. Two different measuring modes are available:

- MEAS: Measurement with deletion of distance-to-go

  Example: `N10 G1 F300 X300 Z200 MEAS=-1`

  Trigger event is the falling edge (-) of probe 1: from deflected to non-deflected status.

- MEAW: Measurement without deletion of distance-to-go

  Example: `N20 G1 F300 X300 Y100 MEAW=1`

  Trigger event is the rising edge of probe 1: from non-deflected to deflected status.

The measurement block is terminated when the probe signal has arrived or the programmed position has been reached. The measurement job can be cancelled with the "RESET" key.

Reference:

Programming and Operating Manual

Note

If a GEO axis (axis in the WCS) is programmed in a measuring block, the measured values are stored for all current GEO axes.

10.3.2 Measurement results

Reading measurement results in the program

The results of the measuring command can be read in the part program via system variables.

- System variable **$AC_MEA[1]**

  Query measurement job status signal.

  The variable is deleted at the beginning of a measurement. The variable is set as soon as the probe fulfills the activation criterion (rising or falling edge). Execution of the measurement job can thus be checked in the part program.

- System variable **$AA_MM[axis]**

  Access to measured value in the machine coordinate system (MCS) Read in part program. [axis] stands for the name of the measurement axis (X, Y, ...).

- System variable **$AA_MW[axis]**

Access to measured value in the workpiece coordinate system. Read in part program. [axis] stands for the name of the measurement axis (X, Y, ...).

References:

Programming and Operating Manual

PLC service display

The measuring signal can be controlled via the "PLC status" menu in the "SYSTEM" → "PLC":

IS "Probe 1 activated" (DB2700.DBX0001.0).

The current measuring status of the axis is shown by the IS "Measurement active" (measuring block with this axis running).

10.4 Measurement accuracy and functional testing

10.4.1 Measuring accuracy

Accuracy

The propagation time of the measuring signal is determined by the hardware used. The delay times are in the µs range plus the probe response time.

The measurement uncertainty is calculated as follows:
Measurement uncertainty = measuring signal propagation time x traversing velocity

Correct results can only be guaranteed for traversing velocity where not more than one triggering signal arrives per position controller cycle.

10.4.2 Probe functional test

Example of functional test

The functional test for the probe is conducted favorably via an NC program.

```
%_N_PRUEF_MESSTASTER_MPF              ;Testing program probe connection
 N10; R10                             ;Flag for trigger status
 N20; R11: messwert in X axis
 N30 T1 D1                            ; Preselect tool offset for probe
 N40 ANF: G0 G90 X0 F150              ;Starting position and meas. velocity
 N50 MEAS=1 G1 X100                   ; Measurement at measuring input 1 in the X
                                      axis
```

```
N60 STOPRE
N70 R10=$AC_MEA[1]                  ; Read switching signal at 1st measuring
                                    input
N80 IF R10==0 GOTOF FEHL1           ;Evaluation of signal
N90 R11=$AA_MW[X]                   ;Read in measured value in workpiece
                                    coordinates
N95 M0
N100 M2
N110 FEHL1: MSG                     ; Probe not switching!
N120 M0
N130 M2
```

Example of repeat accuracy

This program allows the measuring scatter (repeat accuracy) of the entire measuring system (machine-probe-signal transmission) to be calculated.

In the example, ten measurements are taken in the X axis and the measured value recorded in the workpiece coordinates.

It is possible to determine the so-called "random dimensional deviations" which are not subject to any trend.

```
%_N_CHECK_ACCURATE_MPF
  N05; R11                          ; Switching signal
  N06 R12=1                         ; Counter
  N10; R1 to R10                    ; MEAS_VAL_IN _X
  N15 T1 D1                         ; Start conditions, preselect tool offset for
                                    ; probe
  N20 ANF: G0 X0 F150               ;Prepositioning in the measured axis
  N25 MEAS=+1 G1 X100               ; Measurement at 1st measuring input with
                                    ;rising switching edge, in the X axis
  N30 STOPRE                        ; Stop decoding for subsequent evaluation of the
                                    ; result (automatically executed when reading
                                    ; MEA)
  N35 R11= $AC_MEA[1]               ; Read switching signal at 1st measuring input
  N37 IF R11==0 GOTOF FEHL1         ;Check switching signal
  N40 R[R12]=$AA_MW[X]              ;Read measured value in workpiece ;coordinates
  N50 R12=R12+1
  N60 IF R12<11 GOTOB ANF           ;Repeat 10 times
  N65 M0
  N70 M02
  N80 FEHL1: MSG                    ; Probe not switching
  N90 M0
  N95 M02
```

The measurement results R1 to R10 can be read after selecting the parameter display.

10.5 Tool measuring in JOG

Measuring principle

The employed tool is traversed to the probe by the user in the "JOG" mode using the traverse keys or handwheel. The measuring program controls the real measurement sequence with a second approach of the probe and further positioning. In the end the tool offsets are entered.

Advantage: The entered offset values before measuring the tool can deviate entirely from the actual values. The tools must not be "pre-measured".

Note

The tool is "re-measured", not its wear.

Softkeys and templates are provided for use by the user in the "JOG" mode. This supports the user during tool measuring.

Reference:

Programming and Operating Manual

Note

The PLC user program must be created with the necessary sequences. The functionality is not available beforehand.

Extreme caution must be taken when approaching the probe. The probes only have a limited deflection path. They will be damaged or destroyed if this is exceeded! Observe the manufacturers instructions!

In particular, the approach speed should be reduced to such an extent that the probe can always be stopped promptly. "Rapid traverse override" may not be active.

The screen forms provided and the sequence depend on the technology. Accordingly, the following used tool types can be measured:

Milling technology

- Milling tool (geometry length 1 and geometry radius)
- Drill (geometry length 1)

Tool offsets

The screens initially include the active tool T and the active offset number D for the target of the measurement result entry. A different tool can be specified by the PLC via the interface, or the user can enter a different tool T and/or offset number D.

Note

If a tool or an offset number different to the active values has been entered, this must first be made known to the NC for working after measurements have been made with this tool/tool offset, e.g. by programming and start in MDA mode. Only then can the control unit calculate the correct tool offsets.

A **tool length compensation** is automatically entered into the GEO component of the active/specified tool offset D of the active/specified tool, and the associated "wear" and "adapter" components are deleted.

When measuring the **cutter radius** it is assumed that no further offset is applied to the axes of the cutter radius level (values in the axes of the "adapter" component and GEO lengths 2 and 3 are equal to zero). The result for the radius is entered in the "geometry" component. The associated "adapter" and "wear" components of both axes of the level are deleted.

Probe

The tool measuring probe is a touch probe at a fixed location or is swiveled into the working area by means of a mechanical device. If the probe plate is of rectangular design, the edges should be aligned parallel to the axis. The tool/calibration tool is traversed against the measuring probe. The probe must be calibrated before a measurement is taken. This means that the precise probe triggering points in relation to the machine zero are known.

Preparation, probe calibration

- Select the "JOG" mode.
- The following values should be entered in the opened window via the "Settings" softkey:

  return plane, safety clearance, JOG feed, variable increment and direction of rotation of the spindle for general use in JOG and for tool measuring.
- The following value must be entered in the window which opens when pressing the "Data probe" softkey:
  - Feed for automatic probe approach in the measuring program.
  - Probe triggering points (the values are set during calibration).

    If the precise values are known, they can be entered manually. The probe does not then need to be calibrated).
- The adjustment sequence of the probe (calibration) is controlled via the "Measure tool" and "Calibrate probe" softkeys and the opening window. The tool used in this case is the calibration tool with precisely known and entered dimensions.

  The calibration tool for the milling technology is of "cutter" type.

The internal sequence is the same as in measuring. The measuring results, however, are stored in the data for the probe triggering points - not in the tool offsets.

Note

The internal NC programs for measuring or calibrating are configured so that measuring is carried out with the rising edge of the probe.

Measuring sequence

The "JOG" mode is selected. The measuring feed is entered. The probe is calibrated or the precise measuring trigger points are entered.

- Depending on the tool type, the measuring sequence is controlled via the "Measure tool" softkey and further softkeys.
- The IS "Measuring in JOG is active" (DB1700.DBX0003.7) is transmitted to the PLC from the HMI by pressing the "Measure tool" softkey. PLC can specify a different T number to the active one via the IS "T number for tool measuring in JOG" (DB1900.DBD5004). If the probe switches when the selected axis is traversed, NCK outputs the IS "Probe 1 active" (DB2700.DBX0001.0). The PLC then sets IS "Feed disable" (DB3200.DBX0006.0) and NCK stops the axis movement. Feed disable is maintained as long as a traverse key is depressed in JOG and the IS "Measuring in JOG is active" (DB1700.DBX0003.7) is set. After this the PLC outputs the IS "Reset" (DB3000.DBX000.7). The traverse movement in JOG is thereby cancelled.
- HMI recognizes switching of the probe and outputs the change mode to AUTO, IS command "AUTOMATIC mode" (DB1800.DBX0000.0) after the traverse key has been released (immediately after handwheel jog). PLC transfers this to the NCK (DB3000.DBX0000.0).

  AUTO mode is set to active by the NCK (IS "Active mode AUTOMATIC" (DB3100.DBX0000.0)) and is displayed in the HMI screen. PLC cancels the IS "Feed disable (DB3200.DBX0006.0). The HMI then outputs the IS "Mode change disable" (DB1800.DBX0000.4) to the PLC. If the PLC recognizes this signal (is only applied for one PLC cycle), the PLC outputs the IS "Mode change disable" (DB3000.DBX0000.4) to the NCK.

  An NC measuring program has been loaded to the NCK by the HMI. This is activated now. The automatic direction of approach to the probe and the traverse path including the safety clearance is calculated in this measuring program.

  The HMI outputs the command to start the measuring program to the PLC via the IS "Start measuring in JOG" (DB1800.DBX0000.6). The signals in the V1800 area are only applied for a single PLC cycle. The IS "Start measuring in JOG" is therefore stored intermediately in the PLC. The NC measuring program is launched by the PLC by outputting the IS "NC START" (DB3200.DBX0007.1) to the NCK.
- The axis is repositioned by the NC program, the probe is approached again, and finally retracted. The HMI then transmits the command to switch back to the "JOG" mode (DB1800.DBX0000.2) to the PLC. The "Change mode disable" interface signal (DB3000.DBX0000.4) is then reset by the PLC. The PLC outputs the "JOG" mode (DB3000.DBX0000.2) to the NCK and the NCK returns the IS "JOG mode active" (DB3100.DBX0000.2) to the NCK.
- The next direction of approach/axis for traversing to the probe is selected with the "Next step" softkey. The further procedure is analogous - until all directions/axes have been traversed.

After measuring or probe calibration is complete the function can be deselected via the "Back" softkey. This also resets the IS "Measuring in JOG active" (DB1700.DBX0003.0). It is also reset when the operating area is exited. The automatic program can be cancelled via IS "Reset" (DB3000.DBX0000.7) or measuring in JOG can be closed via the "Back" softkey. This also cancels any set IS "Feed disable" (DB3200.DBX0006.0) and IS "Change mode disable" (DB3000.DBX0000.4) or intermediately saved signals.

PLC user program

The required functionality corresponding with the above-described procedure in the PLC user program must be provided by the user.

The toolbox for SINUMERIK 808D supplied by SIEMENS includes a user example in the PLC library. You can use this. In this case it should be noted that PLC_INI (SBR32) and MCP_NCK (SBR38) must always be opened in OB1 as these transfer the signals of the MEAS_JOG (SBR43) subroutine to the NCK/HMI.

10.6 Data table

10.6.1 Machine data

Number	Identifier	Name
General		
13200	MEAS_PROBE_LOW_ACTIVE	Switching characteristics of probe

10.6.2 Interface signals

Number	Bit	Name
HMI signals (from HMI to PLC)		
DB1700.DBX0003	.7	Measuring in JOG active
DB1800.DBX0000	.0	AUTO mode (request by HMI)
DB1800.DBX0000	.1	MDA mode (request by HMI)
DB1800.DBX0000	.2	JOG mode (request by HMI)
DB1800.DBX0000	.4	Change mode disable (request by HMI)
DB1800.DBX0000	.6	Start measuring in JOG (request by HMI)
DB1800.DBX0001	.2	REF machine function (request by HMI)
HMI signals (from PLC to HMI)		
DB1900.DBD 5004		Tool number for tool measuring in JOG (input by PLC)
General (from NCK to PLC)		
DB2700.DBX0001	.0	Probe 1 is actuated
Axis/spindle-specific (from axis to PLC)		
DB390x.DBX0002	.3	Measurement active

EMERGENCY OFF 11

11.1 Brief description

Note

It is the duty of the machine manufacturer to observe national and international standards (see the notes on standards in the following paragraph). The SINUMERIK 808D supports the machine manufacturer in the implementation of the EMERGENCY STOP function in accordance with the specifications in this Description of Functions. The responsibility for the EMERGENCY STOP function (its triggering, execution and acknowledgment) rests exclusively with the machine manufacturer.

Note

Particular reference should be made to the following standards for the EMERGENCY STOP function:

- EN ISO 12100-1
- EN ISO 12100-2
- EN 418
- EN 60204-1

EMERGENCY STOP in the control system

The control system supports the machine manufacturer in implementing an EMERGENCY STOP function on the basis of the following features:

- Activation of EMERGENCY STOP sequence in the NC via a PLC input.
- The EMERGENCY STOP procedure in the NC reduces the speed of all axes and spindles as quickly as possible.
- Unlocking of the EMERGENCY STOP button does not reset the EMERGENCY STOP state. Resetting the control device does not restart the machine.
- After the EMERGENCY STOP state has been cancelled, it is not necessary to reference axes or synchronize spindles (positions are corrected).

EMERGENCY STOP pushbutton

The Siemens machine control panel (MCP) for 808D is equipped with a mushroom-head pushbutton (emergency stop button with one NC and one NO contact each) referred to below as the EMERGENCY STOP pushbutton.

11.2 EMERGENCY STOP sequence

Requirements

Actuation of the EMERGENCY STOP pushbutton or a signal derived directly from the button must be taken to the control system (PLC) as a PLC input. In the PLC user program, this PLC input must be transferred to IS "EMERGENCY STOP" (DB2600.DBX0000.1) in the NC.

Resetting of the EMERGENCY STOP pushbutton or a signal derived directly from the button must be taken to the control system (PLC) as a PLC input. In the PLC user program, this PLC input must be transferred to IS "Acknowledge EMERGENCY STOP" (DB2600.DBX0000.2) in the NC.

Sequence in the NC

The predefined (in EN 418) sequence of internal functions that are implemented to obtain the EMERGENCY STOP state is as follows in the control system:

1. Part program execution is interrupted. All axes and spindles are braked along a braking ramp defined in MD36610 AX_EMERGENCY_STOP_TIME.
2. The IS "808D READY" (DB3100.DBX0000.3) is reset.
3. The IS "EMERGENCY STOP active" (DB2700.DBX0000.1) is set.
4. Alarm 3000 is set.
5. On expiry of a delay that is set for specific axes/spindles in MD36620 SERVO_DISABLE_DELAY_TIME (shutdown delay, controller enable), the controller enable is cancelled. It must be noted that MD36620 must be specified at least as long as MD36610.

Sequence on the machine

The sequence of EMERGENCY STOP functions on the machine is determined solely by the machine manufacturer. Attention should be paid to the following points in connection to the sequence on the NC:

- The sequence of operations in the NC is started with IS "EMERGENCY STOP" (DB2600.DBX0000.1). When the axes and spindles have come to a halt, the power supply must be interrupted, in compliance with EN 418.
- The PLC I/O (digital outputs) are not affected by the sequence in the NC. If individual outputs are required to attain a particular state in the event of an EMERGENCY STOP, the machine manufacturer must include functions for this purpose in the PLC user program.

Note

The interruption of the power feed to the equipment is the responsibility of the machine manufacturer.
If the internal functions in the NC should not be executed in the predetermined sequence in the event of an EMERGENCY STOP, then IS EMERGENCY STOP (DB2600.DBX0000.1) may not be set at any time up to the point that an EMERGENCY STOP defined by the machine manufacture in the PLC user program is reached. As long as the EMERGENCY STOP interface signal has not been set and no other alarm is active, all interface signals are effective in the NC. Any EMERGENCY STOP state defined by the manufacturer can therefore be assumed.

11.3 EMERGENCY STOP acknowledgment

Acknowledge EMERGENCY STOP

The EMERGENCY STOP state is reset only if IS "Acknowledge EMERGENCY STOP" (DB2600.DBX0000.2) followed by IS "Reset" (DB3000.DBX0000.7) are set. It must be noted in this respect that IS "Acknowledge EMERGENCY STOP" and IS "Reset" must be set (together) for a long enough period for IS "EMERGENCY STOP active" (DB2700.DBX0000.1) to be reset (see Fig. 1–1).

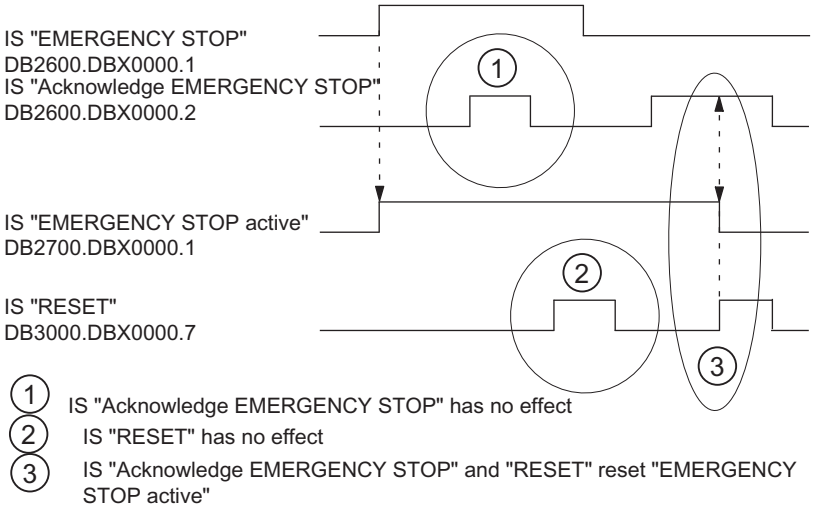


Figure 11-1 Reset emergency stop

Resetting the EMERGENCY STOP state has the following effects:

- IS "EMERGENCY STOP active" is reset.
- The controller enable is switched in.
- IS "Position control active" is set.
- IS "808D READY" is set.
- Alarm 3000 is cleared.
- The part program is aborted.

PLC I/O

The PLC user program must switch the PLC I/O to the correct state for operation of the machine.

Reset

The EMERGENCY STOP state cannot be reset solely by IS "Reset" (DB3000.DBX0000.7) (see diagram above).

Power off/on

Power off/on (POWER ON) cancels the EMERGENCY OFF state unless IS "EMERGENCY OFF" (DB2600.DBX0000.1) is still set.

11.4 Data table

11.4.1 Machine data

Number	Identifier	Name
Axis-specific		
36610	AX_EMERGENCY_STOP_TIME	Length of the braking ramp for error states
36620	SERVO_DISABLE_DELAY_TIME	Shutdown delay controller enable

11.4.2 Interface signals

Number	Bit	Name
General		
DB2600.DBX0000	.0	Braking on the contour with EMERGENCY STOP
DB2600.DBX0000	.1	EMERGENCY STOP
DB2600.DBX0000	.2	Acknowledge EMERGENCY STOP
DB2700.DBX0000	.1	EMERGENCY STOP active
Operating mode signal area		
DB3000.DBX0000	.7	Reset

Reference Point Approach 12

12.1 Fundamentals

Why reference?

The control must be synchronized with the position measurement system of each machine axis so that the control can detect the exact machine zero when it is switched on. This process is known as referencing.

The spindle process (synchronizing) is largely described in Chapter "Spindle (Page 163)".

Position measurement systems

The following position measuring systems can be mounted on the motor:

- Incremental rotary measuring system
- Absolute rotary measuring system

The referencing for the mounted position measuring systems can be set with MD34200 ENC_REFP_MODE (referencing mode).

Output cam

An output cam for referencing may be required for linear axes, and its signal has the following tasks:

- Selection of the direction of travel when approaching the zero mark (synchronized pulse)
- Selection of the zero mark, where required.

BERO

A BERO (inductive proximity switch) can be deployed as the encoder for the synchronized pulse (instead of the zero mark of the position encoder) (preferred for rotary axes, spindles). Here connection is made to the 808D via pin6 (DI3) of terminal X21.

Reference:

Electrical Installation Manual

IS "Active machine function REF" (DB3100.DBX0001.2)

The reference point approach is performed with the REF machine function activated (IS "active machine function REF"). The REF machine function can be selected in JOG modes (IS "REF machine function" (DB3000.DBX0001.2)).

Axis specific referencing

Axis-specific referencing is started separately for each machine axis with the "plus/minus traversing keys" interface signal (DB380x.DBX0004.7 /.6). All axes can be referenced at the same time. If the machine axes are to be referenced in a particular sequence, the following options are available:

- The operator must observe the correct sequence when starting.
- The PLC user program checks the sequence on start-up or defines the sequence itself.
- The order is defined in MD34110 REFP_CYCLE_NR (see channel-specific referencing)

Channel specific referencing

Channel-specific referencing is started with the "activate referencing" interface signal (DB3200.DBX0001.0). The control acknowledges a successful start with IS "Referencing active" (DB3300.DBX0001.0). Each machine axis assigned to the channel can be referenced with channel-specific referencing (this is achieved internally on the control by simulating the plus/minus traversing keys). Axis-specific MD34110 REFP_CYCLE_NR (axis sequence for channel-spec. referencing) can be used to define the sequence in which the machine data is referenced. If all axes entered in MD34110 REFP_CYCLE_NR have reached their end points, the "All axes referenced" interface signal (DB3300.DBX0004.2) is enabled.

Special features

- Referencing is aborted with "Reset" interface signal (DB3000.DBX0000.7). All axes that have not reached their reference point by this time are considered to be not referenced. IS "Referencing active" is reset and alarm 20005 is signaled.
- Working area limiting and software limit switches are not active for non-referenced machine axes.
- The defined axis-specific accelerations are observed at all times during referencing (except when alarms occur).
- The reference point approach can be started only with the direction key for the direction stored in MD34010 REFP_CAM_DIR_IS_MINUS.

Referencing in the part program

One or more axes that have lost their reference can be referenced at the same time. The sequence of the individual phases is identical to axis-specific referencing, except that the process is started with the G74 command instead of the plus/minus traversing keys and is done via the machine axis identifiers.

Reference:

Programming and Operating Manual

Note

MD20700 REFP_NC_START_LOCK = 1 prevents a part program from being started (alarm output) if not all required axes are referenced.

12.2 Referencing with incremental measuring systems

Time sequence

The referencing sequence for incremental measuring systems can be subdivided into three phases:

1. Phase: Traversing to the reference cam
2. Phase: Synchronization with the zero mark
3. Phase: Traversing to the reference point

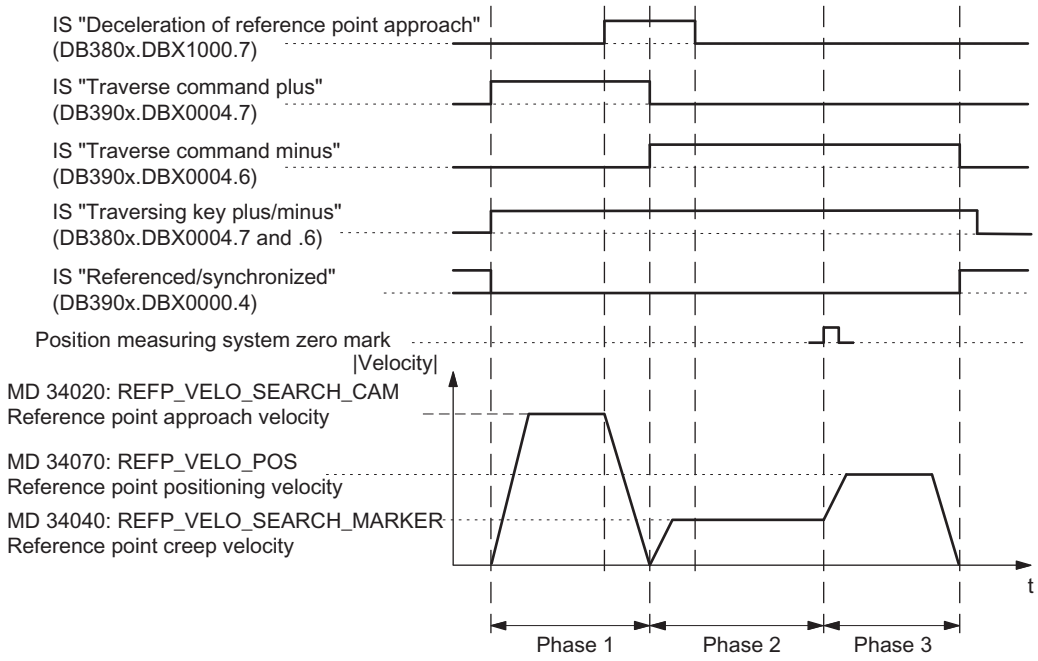


Figure 12-1 Referencing sequence with incremental measuring system (example)

Characteristics of traversing to the reference point cam (phase 1)

- The feedrate override and feedrate stop is in effect.
- The machine axis can be stopped/started.
- The cam must be reached within the traversing distance in MD34030 REFP_MAX_CAM_DIST, otherwise a corresponding alarm is triggered.
- The machine axis must come to a halt at the cam, otherwise a corresponding alarm is triggered.

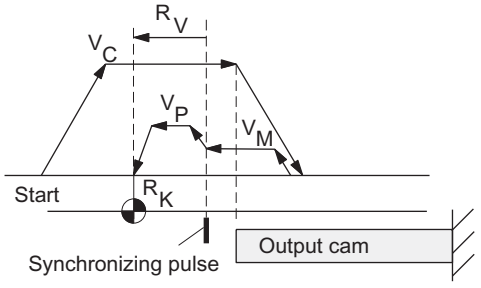
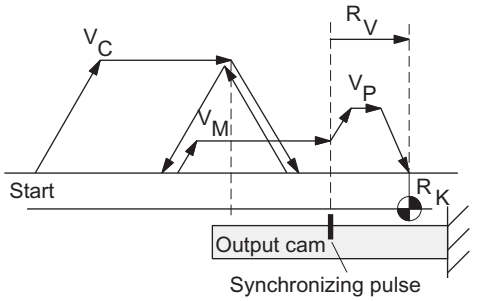
Characteristics when synchronizing with the zero pulse (phase 2)

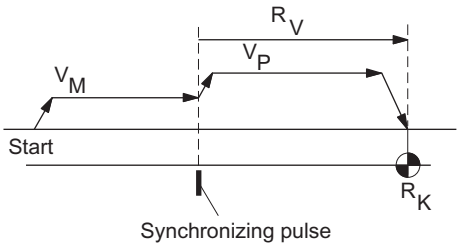
- Feedrate override is not active. Feedrate override 100% is active. Termination occurs at feedrate override 0%.
- Feedrate stop is active, the axis comes to a halt and a corresponding alarm is displayed.
- The machine axis cannot be stopped and restarted with NC stop/NC start.
- Monitoring of the zero mark is active with MD34060 REFP_MAX_MARKER_DIST.

Characteristics of traversing to the reference point (phase 3)

- The feedrate override and feedrate stop is in effect.
- The machine axis can be stopped and re-started with NC stop/NC start.
- If reference point offset is smaller than the braking distance of the machine axis from the reference point positioning velocity to stop, the reference point is approached from the opposite direction.

Different motion sequences during referencing:

Referencing type	Synchronizing pulse (zero mark, BERO)	Motion sequence
With reference cam (MD34000 REFP_CAM_IS_ACTIVE = 1)	Synchronizing pulse before cam, reference coordinate before synchronizing pulse = without reversal: (MD34050 REFP_SEARCH_MARKER_REVERSE = 0)	
	Synchronizing pulse on cam, reference coordinate after synchronizing pulse on cam = with reversal: (MD34050 REFP_SEARCH_MARKER_REVERSE = 1)	

Referencing type	Synchronizing pulse (zero mark, BERO)	Motion sequence
Without reference cam (MD34000 REFP_CAM_IS_ACTIVE = 0)	Reference coordinate after synchronizing pulse	
V_C - reference point approach velocity (MD34020 REFP_VELO_SEARCH_CAM) V_M - reference point creep velocity (MD34040 REFP_VELO_SEARCH_MARKER) V_P - reference point positioning velocity (MD34070 REFP_VELO_POS) R_V - reference point offset (MD34080 REFP_MOVE_DIST + MD34090 REFP_MOVE_DIST_CORR) R_K - reference point coordinate (MD34100 REFP_SET_POS)		

What is the minimum length of a reference cam?

Example of case: Synchronizing pulse before cam, reference coordinate before synchronizing pulse = synchronizing pulse search with falling cam edge.

The reference cam must be long enough, so that when the cam is approached with the reference point approach velocity, the braking operation ends at the cam (the axis comes to a standstill at the cam), and the cam is exited in the opposite direction with the reference point creep velocity (exit with constant velocity).

To calculate the minimum length of the cam, the larger of the two velocities must be inserted into the formula:

$$\text{Min. length} = \frac{(\text{reference point approach speed or creep speed})^2}{2 \times \text{axis acceleration (MD 32300: MAX\_AX\_ACCEL)}}$$

If the machine axis does not come to a halt at the reference cam (interface signal "Reference point approach delay" (DB380x.DBX1000.7) is reset), alarm 20001 is output. Alarm 20001 can occur if the reference cam is too short and the machine axis travels over it when decelerating in phase 1.

If the reference cam extends to the end of travel of the axis, an inadmissible starting point for referencing (after the cam) can be excluded.

Reference cam adjustment

The reference cam must be calibrated exactly. The following factors influence the response time of the control when detecting the reference cam ("Reference point approach delay" interface signal):

- Switching accuracy of the reference cam switch
- Delay of the reference cam switch (NC contact)
- Delay at the PLC input
- PLC cycle time
- Internal processing time

Practice has shown that the signal edge of the reference cam, which is required for synchronizing, is aligned between two synchronized pulses (zero marks). This can be achieved by:

- Set MD34080 REFP_MOVE_DIST = MD34090 REFP_MOVE_DIST_CORR = MD 34100 REFP_SET_POS = 0
- Reference axis
- In JOG mode, traverse the axis to half the path length between the two zero marks. This path is independent of the pitch of the leadscrew S and the gear ratio n (e.g. S=10 mm/rev, n=1:1 produces a path of 5 mm).
- Calibrate the cam switch so that switching is done at exactly this position (IS "Reference point approach delay" (DB380x.DBX1000.7))
- Alternatively, the value of MD34092 REFP_CAM_SHIFT can be changed instead of moving the cam switch.

⚠ WARNING

If the reference cam is not calibrated precisely, an incorrect synchronizing pulse (zero mark) may be evaluated. In this case, the control assumes an incorrect machine zero and moves the axes to incorrect positions. Software limit switches act on incorrect positions and are therefore not able to protect the machine.

12.3 Referencing with distance-coded reference markers

12.3.1 General information

Distance-coded reference markers

Measuring systems with distance-coded reference marks consist of two parallel scale tracks:

- Incremental grating
- Reference mark track

The distance between any two consecutive reference markers is defined. This makes it possible to determine the absolute position of the machine axis when two consecutive reference marks are crossed. For example, if the distance between the reference marks is approx. 10 mm, a traverse path of approx. 20 mm is all that is required to reference the machine axis.

Referencing can be performed from any axis position in the positive or negative direction (exception: end of travel range).

12.3.2 Basic parameter assignment

Linear measuring systems

The following data must be set to parameterize linear measuring systems:

- The absolute offset between the machine zero point and the position of the first reference mark of the linear measuring system:

  MD34090 REFP_MOVE_DIST_CORR (reference point/absolute offset)

  See also below: Determining the absolute offset

- Orientation of the length measuring system (equidirectional or inverse) relative to the machine system coordinate system:

  MD34320 ENC_INVERS (linear measuring system inverse to the machine system)

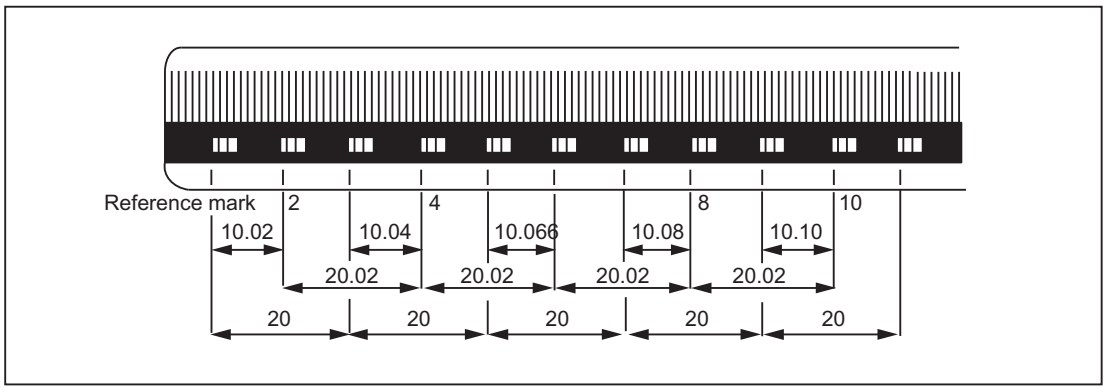


Figure 12-2 DIADUR graduated glass scale with distance-coded reference marks (dimensions in mm for 20 mm scale division)

Rotary measuring system

For rotary measuring systems, the same applies as for linear measuring systems (see above).

Determining the absolute offset

The following procedure is recommended for the determination of the absolute offset between the machine zero point and the position of the first reference mark of a machine axis:

1. Enter the value zero for the absolute offset:

   MD34090 REFP_MOVE_DIST_CORR = 0

2. Perform reference point approach.

   Note

   The reference point should be approached at a point in the machine where the exact position of the machine axis relative to machine zero can be determined easily (using a laser interferometer, for example).

3. Determine the actual position of the machine axis via the operator interface screen.
4. Measure the current position of the machine axis with reference to the machine zero point.
5. Calculate absolute offset and enter in MD34090.

The absolute offset is calculated with respect to the machine coordinate system and depending on the orientation of the measuring system (equidirectional or inverse) as:

Orientation of the measuring system	Absolute offset
equidirectional	Measured position + displayed actual position
Opposite direction	Measured position - displayed actual position

⚠ WARNING

After determining the absolute offset and the entry in MD34090, the reference point traversing for the machine axis must be carried out once more.

Referencing methods

Referencing with distance-coded reference marks can be performed in one of two ways:

- Evaluation of **two** consecutive reference marks:

  MD34200 ENC_REFP_MODE (referencing mode) = 3

  Advantage:

  – Short travel path

- Evaluation of **four** consecutive reference marks:

  MD34200 ENC_REFP_MODE = 8

Advantage:

– Plausibility check by NC is possible
– Increase in reliability of referencing result

12.3.3 Chronological sequence

Chronological sequence

Referencing with distance-coded reference marks can be divided into two phases:

- Phase 1: Travel across the reference marks with synchronization
- Phase 2: Traveling to a fixed destination point

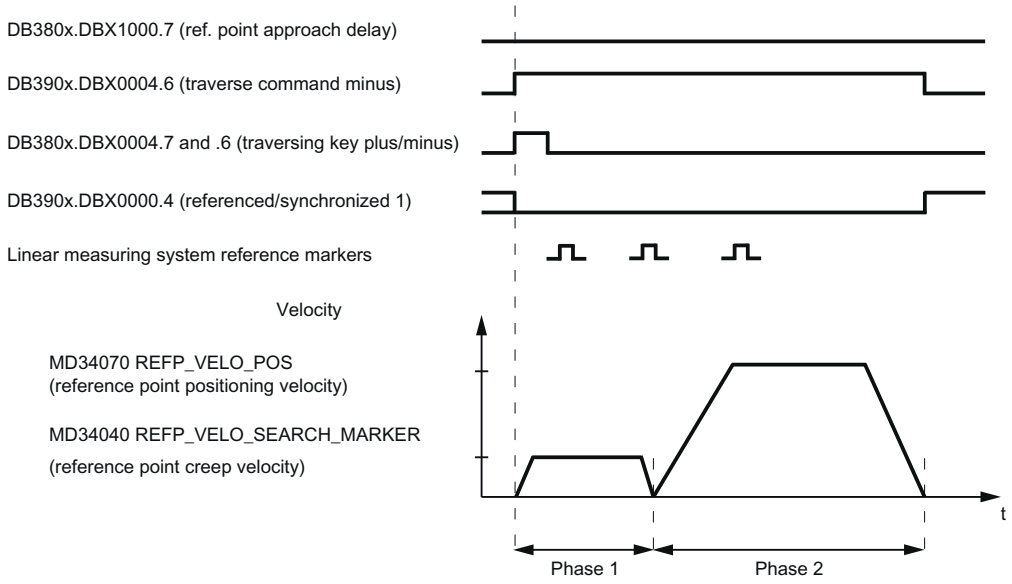


Figure 12-3 Distance-coded reference markers

12.3.4 Phase 1: Travel across the reference marks with synchronization

Phase 1: Start

For information on starting reference point approach, refer to "Axis-specific referencing (Page 147)" and "Channel-specific referencing (Page 147)".

Reference cam

In measuring systems with distance-coded reference marks, reference cams are not required for the actual referencing action. For functional reasons, however, a reference cam is required for channel-specific reference point approach and reference point approach from the part program (G74) before the traversing range end of the machine axis.

Phase 1: Sequence

Sequence without touching a reference cam

Once the reference point approaching process is started, the machine axis accelerates to the reference point shutdown speed set by means of parameter assignment:

MD34040 REFP_VELO_SEARCH_MARKER (reference point shutdown speed)

Once the number of reference markers set by means of parameter assignment has been crossed, the machine axis is stopped again and the actual value system of the machine axis is synchronized to the absolute position calculated by the NC.

Sequence when starting from the reference cam

If the machine axis is at the reference cam at the start of the reference point traversing, it accelerates to the parameterized reference point creep velocity against the parameterized reference point approach direction:

MD34040 REFP_VELO_SEARCH_MARKER (reference point shutdown speed)

MD34010 CAM_DIR_IS_MINUS (reference point approach in minus direction)

That ensures that the machine axis does not reach the travel range limit before it has crossed the parameterized number of reference marks.

Once the number of reference markers set by means of parameter assignment has been crossed, the machine axis is stopped again and the actual value system of the machine axis is synchronized to the absolute position calculated by the NC.

Sequence when contact is made with reference cam during referencing

Once the reference point approaching process is started, the machine axis accelerates to the reference point shutdown speed set by means of parameter assignment:

MD34040 REFP_VELO_SEARCH_MARKER (reference point shutdown speed)

Before the machine axis travels over the parameterized number of reference marks, it touches the reference cam. It is then reversed and reference mark search is restarted in the opposite direction.

Once the number of reference markers set by means of parameter assignment has been crossed, the machine axis is stopped again and the actual value system of the machine axis is synchronized to the absolute position calculated by the NC.

Plausiblity check of the reference mark distance

An error occurs if, during reference point traversing for two subsequent reference marks, the NC determines a distance greater than twice the parameterized reference mark distance.

MD34300 ENC_REFP_MARKER_DIST (reference marker distance)

The machine axis will then traverse in opposite direction at half the parameterized reference point creep velocity (MD34040) and the search for reference mark is restarted.

If a faulty reference mark distance is detected again, the machine axis is stopped and the reference point traversing is aborted (alarm 20003 "fault in the measuring system").

Abort criterion

If the parameterized number of reference marks is not detected within the parameterized distance, the machine axis is stopped and reference point traversing is aborted.

MD34060 REFP_MAX_ MARKER_DIST (max. distance to the reference marker)

Features of phase 1

After phase 1 is successfully completed, the actual value system of the machine axis is synchronized.

12.3.5 Phase 2: Travel to fixed stop

Phase 2: Start

Phase 2 is automatically started when phase 1 has been completed without an alarm.

Initial situation:

- The machine axis is positioned directly behind the last of the parameterized number of reference marks.
- The actual value system of the machine axis is synchronized.

Phase 2: Sequence

In Phase 2, the machine axis completes reference point approach by traversing to a defined target position (reference point). This action can be suppressed in order to shorten the reference point approach:

MD34330 STOP_AT_ABS_MARKER

Value	Meaning
0	Travel to target position
1	No travel to target position

Travel to target position (normal case)

The machine axis accelerates to the parameterized reference point position velocity and travels to the parameterized target point (reference point):

MD34070 REFP_VELO_POS (reference point positioning velocity)

MD34100 REFP_SET_POS (reference point value)

The machine axis is referenced. To identify this, the NC sets the relevant interface signal:
DB390x.DBX0000.4 = 1 (referenced/synchronized 1)

No travel to target position

The machine axis is now referenced. To identify this, the NC sets the relevant interface signal:
DB390x.DBX0000.4 = 1 (referenced/synchronized 1)

Features of phase 2

Phase 2 will display different characteristics, depending on whether a reference point cam is parameterized for the machine axis.

Machine axis without reference point cam

MD34000 REFP_CAM_IS_ACTIVE (axis with reference point cam) = 0

Properties:

- Feed override active.
- The feed stop (channel-specific and axis-specific) is active.
- NC STOP and NC START are active.

Machine axis with reference point cam

MD34000 REFP_CAM_IS_ACTIVE (axis with reference point cam) = 1

Properties:

- Feedrate override is **not** active.

  Machine axis moves internally when feedrate override = 100%.

  If a feedrate override of 0% is specified, an abort occurs.
- The feed stop (channel-specific and axis-specific) is active.
- NC-STOP and NC-START are **not** active.
- If the parameterized number of reference marks is not detected within the parameterized distance after the exit of the reference cam, the machine axis will be stopped.

  MD34060 REFP_MAX_ MARKER_DIST (max. distance to the reference marker)

Special features of rotary measuring systems

On rotary distancecoded measuring systems, the absolute position can only be determined uniquely within one revolution. Depending on the mechanical mounting of the encoder, the overtravel of the absolute position in the hardware does not always coincide with the traversing range of the rotary axis.

Special features of modulo rotary axes

With module rotary axes, the reference point position is mapped on the parameterized modulo range:

MD30330 MODULO_RANGE (magnitude of modulo range)

MD30340 MODULO_RANGE_START (starting position of modulo range)

Note

The reference point position is mapped onto the assigned (ghost) modulo range even with axis function "Determination of reference point position rotary, distance-coded encoder within the configured modulo range":

MD30455 MISC_FUNCTION_MASK (axis functions), Bit1 = 1

12.4 Secondary conditions for absolute encoders

Calibration time

The calibration process determines the offset between the machine zero and the encoder zero and stores it in a non-volatile memory. Normally, calibration need only be performed once, i.e. during first commissioning. The control then knows the value and can calculate the absolute machine position from the encoder absolute value at any time. This status is identified by MD34210 ENC_REFP_STATE=2.

The offset is stored in MD34090 REFP_MOVE_DIST_CORR.

The calibration process must be repeated in the following situations:

- After mounting/removal or replacement of encoder or of motor with built-in encoder.
- After change of an existing gear unit between motor (with absolute encoder) and load.
- Generally speaking, every time the mechanical connection between the encoder and load is separated and not reconnected in exactly the same way.

Note

The control may not always recognize the need for recalibration. If it detects such a need, it sets MD34210 to 0 or 1. The following is detected: changeover to another gear speed with a different gear ratio between the encoder and load.

In all other cases, the user must overwrite MD34210.

Data backup

When machine data is backed up, the status of MD34210 ENC_REFP_STATE is also saved.

By loading this data set, the axis is automatically deemed calibrated!

⚠ WARNING

If the data set has been taken from another machine (e.g. series startup), calibration must still be carried out after loading and activating the data.

12.5 Data table

12.5.1 Machine data

Number	Identifier	Name
Channel-specific		
20700	REFP_NC_START_LOCK	NC-Start disable without reference point
Axis-specific		
30200	NUM_ENCS	Number of encoders
30240	ENC_TYP[0]	Actual value encoder type
30330	MODULO_RANGE	Magnitude of modulo range
30340	MODULO_RANGE_START	Starting position of modulo range
31122	BERO_DELAY_TIME_PLUS[0]	BERO delay time in plus direction
31123	BERO_DELAY_TIME_MINUS[0]	BERO delay time in minus direction
34000	REFP_CAM_IS_ACTIVE	Axis with reference cam
34010	REFP_CAM_DIR_IS_MINUS	Reference point approach in minus direction
34020	REFP_VELO_SEARCH_CAM	Reference point approach velocity
34030	REFP_MAX_CAM_DIST	Maximum distance to reference cam
34040	REFP_VELO_SEARCH_MARKER[0]	Reference point creep speed
34050	REFP_SEARCH_MARKER_REVERSE[0]	Direction reversal to reference cam
34060	REFP_MAX_MARKER_DIST[0]	Maximum distance to reference marker; maximum distance to 2 reference markers with distance-coded scales
34070	REFP_VELO_POS	Reference point positioning velocity
34080	REFP_MOVE_DIST[0]	Reference point distance/destination point for distancecoded system
34090	REFP_MOVE_DIST_CORR[0]	Reference point/absolute offset, distancecoded
34092	REFP_CAM_SHIFT[0]	Electronic reference point cam shift for incremental measuring systems with equidistant zero marks.
34093	REFP_CAM_MARKER_DIST[0]	Reference cam/reference mark distance
34100	REFP_SET_POS[0]...[3]	Reference point value
34110	REFP_CYCLE_NR	Axis sequence for channel-specific referencing
34200	ENC_REFP_MODE[0]	Referencing mode
34210	ENC_REFP_STATE[0]	Status of absolute encoder
34220	ENC_ABS_TURNS_MODULO[0]	Absolute encoder range for rotary encoders
34330	REFP_STOP_AT_ABS_MARKER[0]	Distancecoded linear measuring system without destination point
36300	ENC_FREQ_LIMIT[0]	Encoder frequency limit
36302	ENC_FREQ_LIMIT_LOW[0]	Encoder limit frequency resynchronization
36310	ENC_ZERO_MONITORING[0]	Zero mark monitoring

12.5.2 Interface signals

Number	Bit	Name
Specific to operating mode		
DB3000.DBX0001	.2	Machine function REF
DB3100.DBX0001	.2	Active machine function REF
Channel-specific		
DB3200.DBX0001	.0	Activate referencing
DB3300.DBX0001	.0	Referencing active
DB3300.DBX0004	.2	All axes referenced
Axis-specific		
DB380x.DBX0000	.5	Position measuring system 1
DB380x.DBX0004	.6 and .7	Traversing key minus/plus
DB380x.DBX1000	.7	Reference point approach delay
DB390x.DBX0000	.4	Referenced, synchronizing 1
DB390x.DBX0004	.6 and .7	Traverse command minus/plus

Spindle 13

13.1 Brief description

Spindle functions

Depending on the machine type the following functions are possible for a spindle controlled by the NC:

- Input of a direction of rotation for the spindle (M3, M4)
- Input of a spindle speed (S)
- Spindle stop, without orientation (M5)
- Spindle positioning (SPOS=) (position-controlled spindle required)
- Gear change (M40 to M45)
- Thread cutting/tapping (G33, G34, G35, G331, G332, G63)
- Revolutional feedrate (G95)
- Constant cutting rate (G96)
- Position encoder assembly on the spindle or on the spindle motor
- Spindle monitoring for min. and max. speed
- Dwell time in spindle revolutions (G4 S)

An "enabled" spindle can be used instead of a controlled spindle. However, a spindle speed (S) is then **not** entered via the program but, for example, manually (gearbox) at the machine. This does not permit programming of speed limits. The following is possible via the program:

- Input of a direction of rotation for the spindle (M3, M4)
- Spindle stop, without orientation (M5)
- Tapping (G63)

If the spindle has a position encoder, the following functions are also available:

- Thread cutting/tapping (G33, G34, G35)
- Revolutional feedrate (G95)

If the spindle is enabled, the setpoint output for the spindle via MD30130 CTRLOUT_TYPE = 0 must be suppressed.

Definition of the spindle

A machine axis is declared a spindle by setting the following machine data:

- MD30300 IS_ROT_AX
- MD30310 ROT_IS_MODULO
- MD30320 DISPLAY_IS_MODULO
- MD35000 SPIND_ASSIGN_TO_MACHAX.

The IS "Spindle/no axis" reports the spindle mode (DB390x.DBX0000.0).

13.2 Spindle modes

13.2.1 Spindle modes

Spindle modes

The spindle can have the following modes:

- Control mode, see Section "Spindle control mode (Page 165)"
- Oscillation mode, see Section "Spindle oscillation mode (Page 166)"
- Positioning mode, see Section "Spindle positioning mode (Page 168)"
- Axis mode
- Tapping without compensating chuck, see also Chapter "Feed (Page 187)"

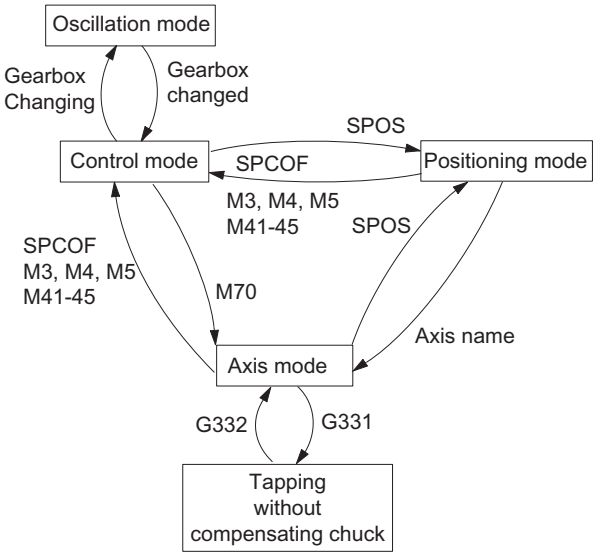


Figure 13-1 Switching between spindle modes

Switching between spindle modes

- Control mode ---> oscillation mode

  The spindle changes to oscillation mode if a new gear stage has been specified using automatic gear stage selection (M40) in conjunction with a new S value or by M41 to M45. The spindle only changes to oscillation mode if the new gear stage is not equal to the current actual gear stage.

- Oscillation mode ---> control mode

  When the new gear stage is engaged, the IS "Oscillation mode" (DB390x.DBX2002.6) is reset and the spindle is switched to control mode with the IS "Gear changed" (DB380x.DBX2000.3). The last programmed spindle speed (S value) is reactivated.

- Control mode ---> positioning mode

  To stop the spindle from rotation (M3 or M4) with orientation or to reorient it from standstill (M5), SPOS, SPOSA or M19 are used to switch to positioning mode.
- Positioning mode ---> control mode

  SPCOF, M3, M4, M5, or M41-45 are used to change to control mode if the orientation of the spindle is to be terminated. The last programmed spindle speed (S value) is reactivated.
- Positioning mode ---> oscillation mode

  If the orientation of the spindle is to be terminated, M41 to M45 can be used to change to oscillation mode. When the gear change is complete, the last programmed spindle speed (S value) and M5 (control mode) are reactivated.
- Positioning mode ---> tapping without compensation chuck

  Tapping without compensation chuck (thread interpolation) is activated via G331/G332. SPOS must first be used to set the spindle to position-controlled operation.

13.2.2 Spindle control mode

When control mode?

The spindle is in control mode with the following functions:

- Constant spindle speed S, M3/M4/M5 and G94, G95, G97, G33, G63
- Constant cutting rate G96 S, M3/M4/M5

Requirements

A spindle position actual value sensor is absolutely essential for M3/M4/M5 in conjunction with revolution feedrate (G95, F in mm/rev or inch/rev), constant cutting rate (G96, G97), thread cutting (G33).

Independent spindle reset

MD35040 SPIND_ACTIVE_AFTER_RESET defines the response of the spindle after reset or program end (M2, M30):

- If MD value=0, the spindle is immediately braked to rest at the valid acceleration. The last programmed spindle speed and direction of rotation are deleted.
- If MD value = 1 (independent spindle reset), the last programmed spindle speed (S function) and the last programmed direction of spindle rotation (M3, M4, M5) are retained. If prior to reset or end of program the constant cutting speed (G96) is active, the current spindle speed (in relation to 100% spindle override) is internally accepted as the spindle speed last programmed.

Note

The spindle can always be stopped with the IS "Delete distance-to-go / Spindle Reset". **CAUTION:** The program continues at G94! With G95 the axes stop due to the missing feedrates as does the program run if G1, G2, ... is active.

13.2.3 Spindle oscillation mode

Starting oscillation mode

This oscillation movement makes it easy to engage a new gear stage. In principle, the new gear stage can also be engaged without oscillation

The spindle is in oscillation mode if a new gear stage was defined using the automatic gear stage selection (M40) or M41 to M45 (IS "Change gear" (DB390x.DBX2000.3) is enabled). The IS "Change gear" is only enabled when a new gear stage is selected that is not the same as the current actual gear stage. The spindle oscillation is started with the IS "Oscillation speed" (DB380x.DBX202.5).

If the IS "Oscillation speed" is enabled without defining a new gear stage, the spindle does not change to oscillation mode.

Oscillation is started with the IS "Oscillation speed". The setting of the IS "Oscillation via PLC" (DB380x.DBX2002.4) distinguishes between:

- Oscillation via NCK
- Oscillation via PLC

Oscillation time

The oscillation time for oscillation mode can be defined in a machine date for each direction of rotation:

- Oscillation time in M3 direction (referred to as t1 in the following):

  MD35440 SPIND_OSCILL_TIME_CW
- Oscillation time in M4 direction (referred to as t2 in the following):

  MD35450 SPIND_OSCILL_TIME_CCW

Oscillation via NCK

Phase 1: With the IS "Oscillation speed" (DB380x.DBX2002.5) , the spindle motor accelerates to the velocity (with oscillation acceleration) specified in MD35400 SPIND_OSCILL_DES_VELO (oscillation speed). The starting direction is defined by MD35430 SPIND_OSCILL_START_DIR (starting direction during oscillation).

Phase 2: If time t1 (t2) has elapsed, the spindle motor accelerates in the opposite direction to the speed defined in MD35400 SPIND_OSCILL_DES_VELO (oscillation speed). Time t2 (t1) starts.

Phase 3: When time t2 (t1) expires, the spindle motor accelerates in the opposite direction (same direction as phase 1), etc.

Oscillation via PLC

With the IS "Oscillation speed" (DB380x.DBX2002.5) , the spindle motor accelerates to the velocity (with oscillation acceleration) specified in MD35400 SPIND_OSCILL_DES_VELO (oscillation speed).

The direction of rotation is determined by IS "Set direction of rotation counterclockwise" or IS "Set direction of rotation clockwise" (DB380x.DBX2002.7 or .6).

The oscillation movement and the two times t1 and t2 (for clockwise and counterclockwise rotation) must be simulated on the PLC.

End of oscillation mode

The IS "Gear changed" (DB380x.DBX2000.3) informs the NC that the new gear stage (IS "Actual gear stage" (DB380x.DBX2000.0 to .2)) applies and oscillation mode is exited. The actual gear stage should correspond to the set gear stage. Oscillation mode is also ended if the IS "oscillation speed" (DB380x.DBX2002.5) is still set. The last programmed spindle speed (S function) and spindle rotation (M3, M4 or M5) are active again.

After termination of oscillation mode the spindle returns to control mode.

All gear specific limit values (min./max. speed etc.) correspond to the set values of the actual gear stage and are deactivated when the spindle stops.

Block change

If the spindle has been changed over to oscillation mode, IS "Change gear" (DB390x.DBX2000.3) is set, part program processing is stopped. A new block is not executed. If oscillation mode is terminated using the IS "Gear switched" (DB380x.DBX2000.3), the execution of the part program is continued. A new block is executed.

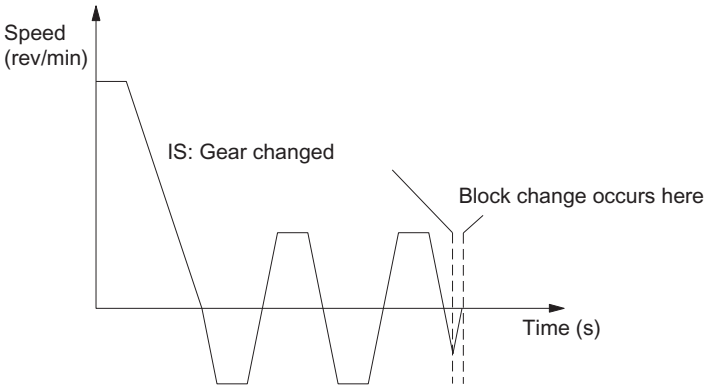


Figure 13-2 Block change following oscillation mode

Special features

- The acceleration is defined by MD35410 SPIND_OSCILL_ACCEL (acceleration during oscillation).
- If the IS "oscillation speed" (DB380x.DBX2002.5) is reset, the oscillation stops. However, the spindle **remains** in oscillation mode.
- The IS "Gear changed" should always be used for terminating gear stage change.
- The IS "Reset" (DB3000.DBX0000.7) does **not** terminate oscillation mode.
- If an indirect measuring system is used, synchronization is lost. The spindle is re-synchronized the next time the zero mark is crossed.

Reset during gear stage change

The spindle cannot be stopped via IS "Reset" (DB3000.DBX0000.7) or IS "NC Stop" (DB3200.DBX0007.3) if the spindle is in oscillation mode for gear stage change and the IS "Gear changed" (DB380x.DBX2000.3) is not yet available.

In this case, alarm 10640 "Stop not possible during gear change" is displayed if reset is selected. After changing the gear stages, the reset request is performed and the alarm cleared, if this is still present at the interface.

Note

Option for aborting: Set IS "Delete distance-to-go / Spindle Reset" (DB380x.DBX0002.2).

13.2.4 Spindle positioning mode

When is positioning mode used?

The spindle positioning mode stops the spindle at the defined position and activates the position control, which remains active until it is deactivated. With the **SPOS =**..... program function, the spindle is in positioning mode (see also Section "Programming (Page 179)").

Block change

The block change is carried out when all functions programmed in the block have reached their end criterion (e.g. axis traverse completed, all auxiliary functions acknowledged by PLC) **and** the spindle has reached its position (IS "Exact stop fine" for spindle (DB390x.DBX0000.7)).

Requirements

A spindle position actual value encoder is absolutely essential.

Positioning from rotation

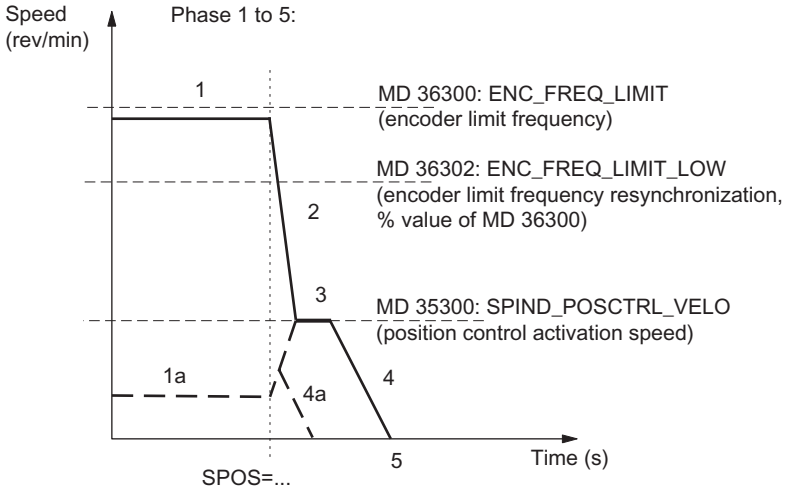


Figure 13-3 Positioning from rotation at different speeds

Sequence

Phase 1: Spindle rotates at a lower speed than the encoder limit frequency. The spindle is synchronized. It is set to control mode. Process continues with Phase 2.

Phase 1a: Spindle rotates at a lower speed than the position control activation speed. The spindle is synchronized. It is set to control mode. The rest of the sequence is possible via 4a.

Phase 1b (not shown): Spindle rotates at a speed higher than the encoder limit frequency. The spindle is not synchronized initially, but is then synchronized when the rotation speed falls below the speed defined by the encoder frequency in MD36302 ENC_FREQ_LIMIT_LOW (% value of MD36300). Sequence continues with Phase 2.

Phase 2: When the SPOS command takes effect, the spindle starts to decelerate with the acceleration stored in MD35200 GEAR_STEP_SPEEDCTRL_ACCEL until it reaches the position control activation speed.

Phase 3: When the position-control activation speed stored in MD35300 SPIND_POSCTRL_VELO is reached:

- The position control is activated.
- The distance-to-go (to target position) is calculated. (easier from Phase 1a)
- The acceleration is switched to MD35210 GEAR_STEP_POSCTRL_ACCEL. (acceleration in position control mode) (always active below the position control activation speed)

Phase 4: The spindle brakes from the calculated "braking point" with MD35210 GEAR_STEP_POSCTRL_ACCEL to the target position.

Phase 5: The position control remains active and stops the spindle in the programmed position. The IS "Position reached with exact stop fine" (DB390x.DBX0000.7) and "... coarse" (DB390x.DBX0000.6) are set if the distance between the spindle actual position and the programmed position (spindle setpoint position) is less than the settings for the exact stop fine and coarse limits (respectively defined in MD36010 STOP_LIMIT_FINE and MD36000 STOP_LIMIT_COARSE).

Positioning from standstill, spindle not synchronized

The spindle is not synchronized after the control has been activated. The first movement of the spindle must be positioning (SPOS=...).

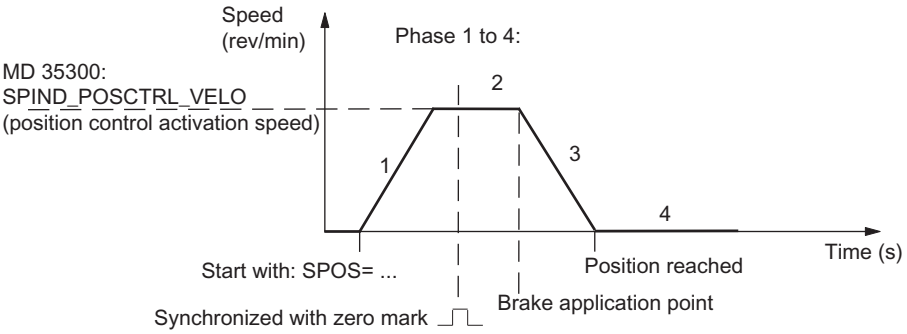


Figure 13-4 Positioning with stopped, non-synchronized spindle

Sequence

Phase 1: Programming SPOS accelerates the spindle with the acceleration in MD35210 GEAR_STEP_ POSCTRL_ACCEL (acceleration in position control mode) until the maximum speed entered in MD35300 SPIND_POSCTRL_VELO (position control activation speed) is reached.

The direction of rotation is defined by MD35350 SPIND_POSITIONING_ DIR (direction of rotation during positioning from standstill), if no input results from SPOS programming (ACN, ACP, IC). The spindle is synchronized with the next zero mark of the position actual value encoder.

Phase 2: When the spindle is synchronized, the position control is activated. The spindle rotates at the maximum speed stored in MD35300 SPIND_POSCTRL_VELO until the braking start point calculation identifies the point at which the programmed spindle position can be approached accurately with the defined acceleration.

Phase 3: At the brake application point, the spindle is braked down to standstill with the acceleration set in MD35210 GEAR_STEP_ POSCTRL_ACCEL (acceleration in position control mode).

Phase 4: The spindle has reached the target point and is stationary. The position control is active and stops the spindle in the programmed position. The IS "Position reached with exact stop fine" (DB390x.DBX0000.7) and "... coarse" (DB390x.DBX0000.6) are set if the distance between the spindle actual position and the programmed position (spindle setpoint position) is less than the settings for the exact stop fine and coarse limits (MD36010 STOP_LIMIT_FINE and MD36000 STOP_LIMIT_COARSE).

Positioning from standstill, spindle is synchronized

The spindle has already been turned by one spindle revolution with M3 and M4 and was then brought to a standstill with M5.

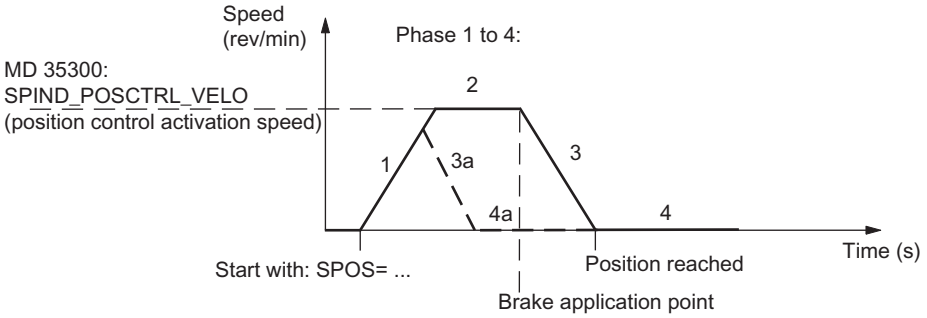


Figure 13-5 Positioning with stationary, synchronized spindle

Sequence

The spindle travels to the programmed end point optimally in terms of time. Depending on the appropriate secondary conditions, the operational sequences in phases 1 - 2 - 3 - 4 or 1 - 3a - 4a are executed.

Phase 1: SPOS will switch the spindle to position control mode. The acceleration from MD35210 GEAR_STEP_POSCTRL_ACCEL (acceleration in the position control mode) is activated. The direction of rotation is determined by the relevant distance-to-go (type of path setting with SPOS).

The speed entered in MD35300 SPIND_POSCTRL_VELO (position control activation speed) is not exceeded. The traversing path to the end point is calculated. If the end point can be accessed immediately from this phase, Phase 3a, 4a continues instead of Phase 2.

Phase 2: Acceleration has been performed up to the speed set in MD35300 SPIND_POSCTRL_VELO (position control activation speed). The brake application point calculation identifies when the programmed spindle position (SPOS=...) can be approached with the acceleration defined in MD35210 GEAR_STEP_POSCTRL_ACCEL.

Phase 3 and Phase 4: The sequence for "Deceleration" and "Position reached" is the same as for non-synchronized spindles.

Spindle reset

The positioning process can be aborted with the IS "Delete distance-to-go/spindle reset" (DB380x.DBX0002.2). However, the spindle remains in positioning mode.

Notes

- In positioning mode the spindle speed override switch continues to be valid.
- Positioning (SPOS) is cancelled with "Reset" or "NC stop".

13.3 Synchronization

Why synchronize?

The control must be synchronized with the position measurement system on the spindle so that the control knows the exact 0 degree position when switched on. Only a synchronized spindle is capable of thread cutting or positioning.

For axes, this process is referred to as referencing, see Chapter "Reference Point Approach".

Installation position of the position measurement system

- Directly on the motor in combination with a BERO proximity switch on the spindle (zero mark encoder)
- Directly on the spindle
- Above the measuring gearbox plus BERO switch on the spindle.

Synchronization possibilities

When the spindle is switched on, the spindle can be synchronized as follows:

- The spindle is started with a spindle speed (S function) and a spindle rotation (M3 or M4), and synchronized with the next zero mark of the position measurement system or with the BERO signal. The 0 degree position is shifted by

  MD34080 REFP_MOVE_DIST + MD34090 REFP_MOVE_DIST_CORR -

  MD34100 REFP_SET_POS.

  Note

  Only use MD34080 for shifting the 0 degree position. Monitoring with MD34060 REFP_MAX_MARKER_DIST should be set to two spindle revolutions (720 degrees).

- Programming SPOS=... from various states (refer to Section "Spindle positioning mode (Page 168)")
- In JOG mode, the spindle is started in speed control mode with the direction keys and synchronizes with the next zero mark of the position measurement system or the BERO signal.

Value acceptance

When synchronizing the spindle, the associated reference point from MD34100 REFP_SET_POS[0] (default value = 0) is transferred and a possible shift of the reference point. These shifts (machine data) act irrespective of the connected measurement system and are described in Chapter "Reference Point Approach".

Maximum encoder frequency exceeded

When the spindle speed reaches a speed (large S value programmed), which exceeds the maximum encoder limit frequency MD36300 ENC_FREQ_LIMIT (the maximum mechanical speed limit of the encoder must not be exceeded), the synchronization is lost. The spindle continues to rotate, but with reduced functionality.

If a speed is then reached that is below the encoder limit frequency in MD36302 ENC_FREQ_LIMIT_LOW (% value of MD36300), the spindle automatically synchronizes with the next zero mark signal. You can achieve this by programming a lower S value, changing the spindle speed override switch, etc.

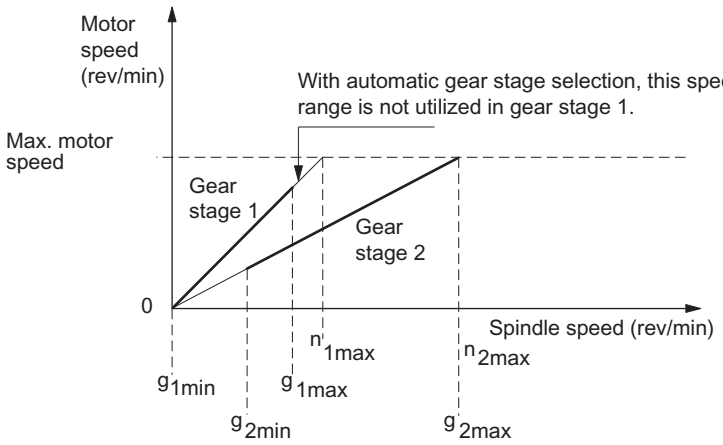
Re-synchronizing

In the following case, however, the position measuring system must be re-synchronized: the position measurement encoder is on the motor, a BERO (distance sensor for synchronization signals) is mounted to the spindle and the gear stage is changed. The synchronization is triggered internally when the spindle is rotating in the new gear stage.

13.4 Gear stage change

Number of gear stages

Five gear stages can be configured for the spindle. If the spindle motor is mounted on the spindle directly (1:1) or with a non-adjustable gear ratio, MD35010 GEAR_STEP_CHANGE_ENABLE (gear stage change is possible) must be set to zero.



Definable by MD:
- n1max ... max. spindle speed of the 1st gear stage
- g1min ... min. spindle speed of the 1st gear stage for autom. gear stage selection
- g1max ... max. spindle speed of the 1st gear stage for autom. gear stage selection
- n2max ... max. spindle speed of the 2nd gear stage
- g2min ... min. spindle speed of the 2nd gear stage for autom. gear stage selection
- g2max ... max. spindle speed of the 2nd gear stage for autom. gear stage selection

Figure 13-6 Gear stage change with gear stage selection

Defining a gear stage

A gear stage can be defined as follows:

- Permanent definition in the part program (M41 to M45)
- Automatic definition by the programmed spindle speed (M40)

In the case of M40, the spindle must be in control mode for automatic gear stage selection with an S value. The gear stage change is otherwise rejected and alarm 22000 "Gear change not possible" is output.

M41 to M45

The gear stage can be permanently defined in the part program with M41 to M45. If a gear stage is defined by M41 to M45, which is different than the current (actual) gear stage, the IS "Change gear" (DB390x.DBX2000.3) and the IS "Set gear stage A" to "...C" (DB390x.DBX2000.0 to .2) are set. The programmed spindle speed (S) then refers to this permanently defined gear stage. If a spindle speed exceeding the maximum speed of the permanently defined gear stage is programmed, the speed is limited to the maximum speed of this gear stage and the IS "Programmed speed too high" (DB390x.DBX2001.1) is enabled. If a speed is programmed lower than the minimum speed of this gear stage, the speed is raised to this speed. The IS "Setpoint speed increased" (DB390x.DBX2001.2) is then enabled.

M40

M40 in the part program causes the gear stage to be selected automatically by the control. The control checks which gear stage is possible for the programmed spindle speed (S function). If the suggested gear stage is not equal to the current (actual) gear stage, the IS "Change gear" (DB390x.DBX2000.3) and the IS "Set gear stage A to C" (DB390x.DBX2000.0 to .2) are enabled.

The automatic gear stage selection function initially compares the programmed spindle speed with the minimum and maximum speed of the current gear stage. If the comparison is positive, a new gear stage is not defined. If the comparison is negative, the comparison is performed on each of the gear stages (starting with gear stage 1) until the result is positive. If the comparison in the 5th gear stage is also not positive, no gear stage change is triggered. If necessary the speed is limited to the maximum speed of the current gear stage or increased to the minimum speed of the current gear stage, and the IS "Setpoint speed limited" (DB390x.DBX2001.1) or IS "Setpoint speed increased" (DB390x.DBX2001.2) is enabled.

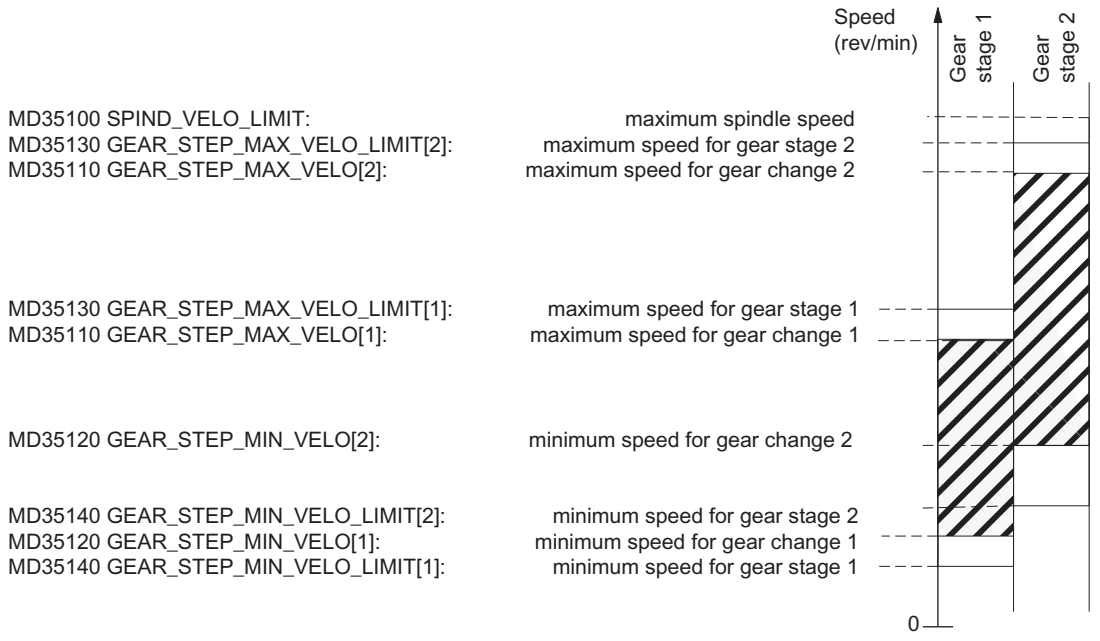


Figure 13-7 Example for speed ranges for automatic gear stage selection (M40)

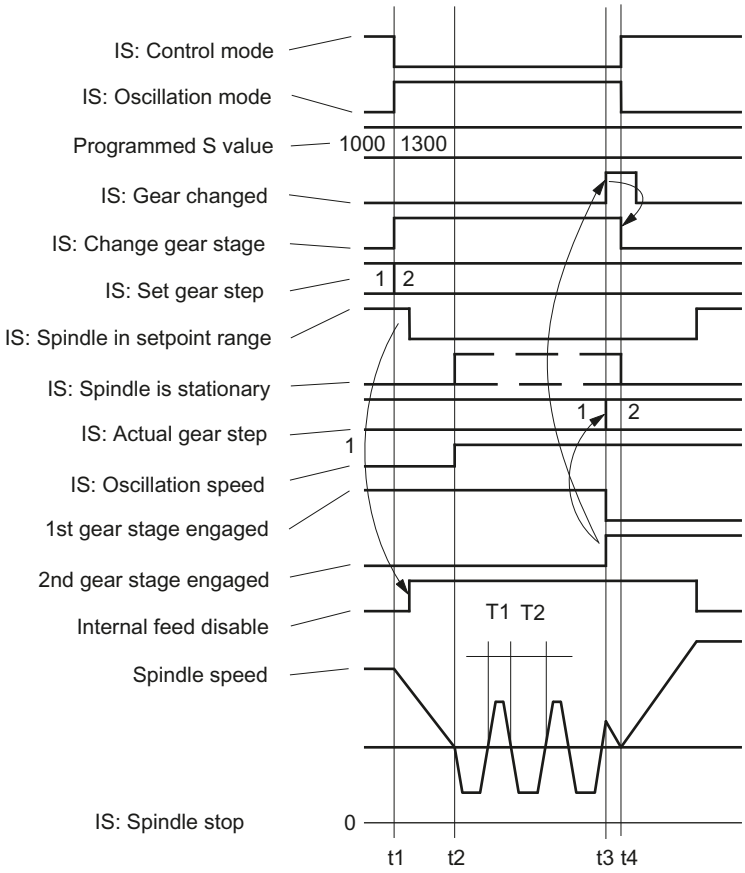
Gear stage change

A new gear stage can only be selected when the spindle is stationary.
The spindle is stopped internally in the control if a gear stage change is requested. If the new gear stage is preselected by M40 and spindle speed or M41 to M45, the IS "Set gear stage A" to "...C" (DB390x.DBX2000.0 to .2) and the IS "Change gear" (DB390x.DBX2000.4) are set. At the point when the IS "Oscillation speed" (DB380x.DBX2002.5) is enabled, the spindle decelerates to a stop with the acceleration for oscillation or with the acceleration for speed control / position control.

The next block in the part program **after** the gear stage change via M40 and S value or M41 to M45 is not performed (same effect as if the IS "Read-in disable" (DB3200.DBX0006.1) were enabled).

When stationary the spindle (IS "Axis/spindle stationary" (DB390x.DBX0001.4)) can be activated with the IS "Oscillation speed" (DB380x.DBX2002.5) (see Section "Spindle oscillation mode (Page 166)"). When the new gear stage is engaged, the PLC user sets the IS "Actual gear stage" (DB380x.DBX2000.0 to .2) and IS "Gear changed" (DB380x.DBX2000.3). The gear stage change is considered completed (spindle mode "Oscillation mode" is deselected) and the spindle is switched to the parameter block of the new actual gear stage. The spindle accelerates at the new gear stage to the spindle speed last programmed (if M3 or M4 are active). The IS "Change gear" (DB390x.DBX2000.3) is reset by the NCK, which causes the PLC user to reset the IS "Gear changed" (DB380x.DBX2000.3). The next block in the part program can be executed.

Typical time sequence for the gear stage change:



t1 When S1300 is programmed, the NCK detects a new gear stage (2nd gear stage), enables IS: Change gear and inhibits execution of the next part program block.

t2 The spindle is stationary and oscillation starts (oscillation via NCK). The Oscillation speed interface signal should be enabled by the time t2.

t3 The new gear stage is engaged. The PLC user transmits the new (actual) Gear stage to the NCK and sets the IS: gear changed.

t4 At this point, the NCK cancels the Change gear interface signal, terminates oscillation, enables execution of the next part program block and accelerates the spindle to the new S value (S1300).

Figure 13-8 Gear stage change with stationary spindle

Parameter set

One parameter set each is provided for each of the five gear stages. The appropriate parameter set is activated through the IS "Actual gear stage A" to "...C" (DB380x.DBX2000.0 to .2).
It is assigned as follows:

Index n	PLC interface, CBA coding	Data of the data set	Contents
0	-	Data for axis mode	Servo gain factor, monitoring functions, speed, acceleration, etc.
1	000 001	Data for 1st gear stage	
2	010	Data for 2nd gear stage	
3	011	Data for 3rd gear stage	
4	100	Data for 4th gear stage	
5	101	Data for 5th gear stage	

The machine data included in a parameter set are marked specifically in Section "Machine Data (Page 184)". The following machine data is added per gear stage for each parameter set index n (n=1 -> 1st gear stage of the spindle, etc.):

- MD35110 GEAR_STEP_MAX_VELO[n]
- MD35120 GEAR_STEP_MIN_VELO[n]
- MD35130 GEAR_STEP_MAX_VELO_LIMIT[n]
- MD35140 GEAR_STEP_MIN_VELO_LIMIT[n]
- MD35200 GEAR_STEP_SPEEDCTRL_ACCEL[n]
- MD35210 GEAR_STEP_POSCTRL_ACCEL[n]
- MD35310 SPIND_POSIT_DELAY_TIME[n]

13.5 Programming

Functions

The spindle can be set for the following functions:

- G95 Revolutional feedrate
- G96 S... LIMS=... Constant cutting rate in m/min, upper speed limit
- G97 Cancel G96 and freeze last spindle speed
- G33, G331, G332 Thread cutting, tapping
- G4 S ... Dwell time in spindle revolutions

M3	CW spindle rotation
M4	CCW spindle rotation
M5	Spindle stop, without orientation
S...	Spindle speed in rpm, e.g. S300
SPOS=...	Spindle positioning, e.g. SPOS=270 -> at position 270 degrees. The block change is only performed when the spindle is in position.
SPOS=DC(Pos)	The direction of motion is retained for positioning while in motion and the position approached. When positioning from standstill, the position is approached via the shortest path.
SPOS=ACN(Pos)	The position is always approached with negative direction of motion. If necessary, the direction of motion is inverted prior to positioning.
SPOS=ACP(Pos)	The position is always approached with positive direction of motion. If necessary, the direction of motion is inverted prior to positioning.
SPOS=IC(Pos)	The traversing path is specified. The direction of traversing is obtained from the sign in front of the traversing path. If the spindle is in motion, the direction of traversing is inverted if necessary to allow traversing in the programmed direction.
M40	Automatic gear stage selection for the spindle
M41 to M45	Select gear stage 1 to 5 for the spindle
SPCON	Position control on
SPCOF	Position control off
M70	Position control on
LIMS=...	Programmable maximum spindle speed for G96

Reference:

Programming and Operating Manual

13.6 Spindle monitoring

13.6.1 Spindle monitoring

Speed ranges

The spindle monitoring functions and the currently active functions (G94, G95, G96, G33, G331, G332, etc.) define the admissible speed ranges of the spindle.

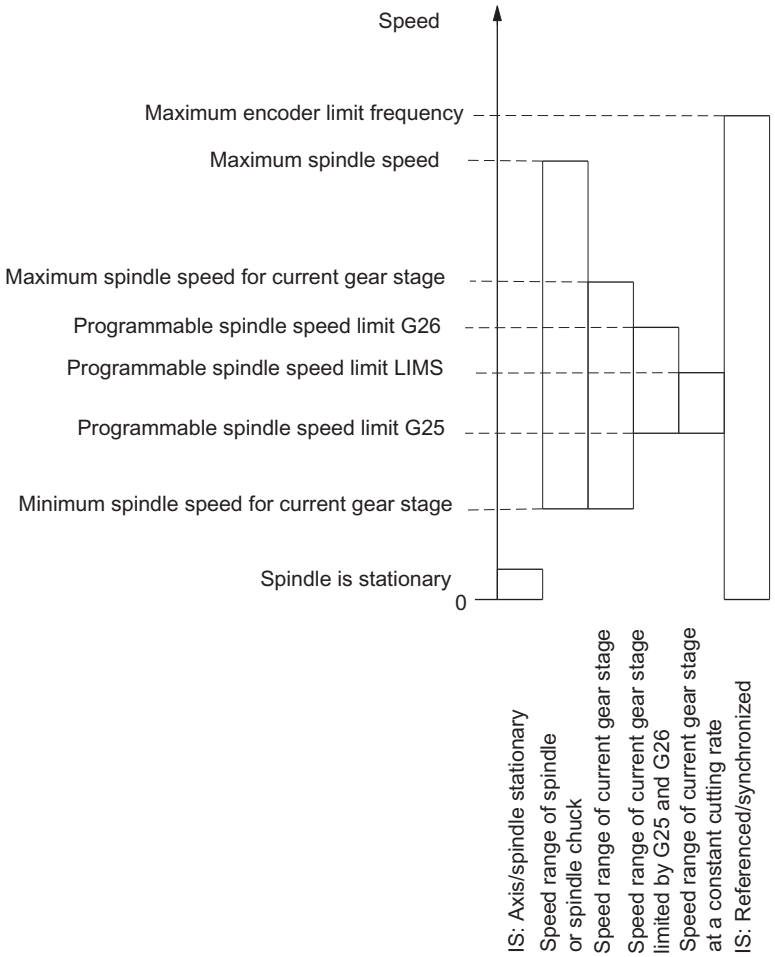


Figure 13-9 Ranges of spindle monitoring functions / speeds

13.6.2 Axis/spindle stationary

Only when the spindle is stationary, i.e. the actual spindle speed is below a value defined in MD36060 STANDSTILL_VELO_TOL, is IS "Axis/spindle stationary" (DB390x.DBX0001.4) set. Functions such as tool change, open machine door, path feed can be activated using the PLC user program.

Monitoring is effective in the three spindle modes.

13.6.3 Spindle in setpoint range

The "Spindle in setpoint range" monitor checks whether the programmed spindle speed has been reached, whether the spindle is stationary (IS "Axis/spindle stationary") or whether it is still in the acceleration phase.

In the spindle "control mode", the speed setpoint (programmed speed with spindle override including the active limits) is compared with the actual speed. If the deviation of the actual speed from the speed setpoint is greater than the spindle speed tolerance set in MD35150 SPIND_DES_VELO_TOL:

- IS "Spindle in setpoint range" (DB390x.DBX2001.5) is set to zero.
- The next machining block is not enabled if

  MD35500 SPIND_ON_SPEED_AT_IPO_START is set.

13.6.4 Maximum spindle speed

Maximum spindle speed

A maximum speed is defined for "maximum spindle speed" spindle monitoring, which the spindle may not exceed.

The maximum spindle speed is entered in MD35100 SPIND_VELO_LIMIT.

The control limits an excessive spindle speed setpoint to this value. If the actual spindle speed exceeds the maximum spindle speed despite allowance for the spindle speed tolerance (MD35150 SPIND_DES_VELO_TOL), there is a drive fault and IS "Speed limit exceeded" (DB390x.DBX2002.0) is set. Furthermore the alarm 22100 is output and all axes and the spindle are decelerated.

13.6.5 Minimum/maximum speed for gear stage

Max. speed

MD35130 GEAR_STEP_MAX_VELO_LIMIT defines the maximum speed for the gear stage. In the gear stage engaged, this set speed can never be exceeded. When the programmed spindle speed is limited, the IS "Set speed limited" (DB390x.DBX2001.1) is enabled.

Minimum speed

MD35140 GEAR_STEP_MIN_VELO_LIMIT defines the minimum speed for the gear stage. It is not possible that the speed falls below this (set) speed if an S value is programmed, which is too small. Then, the IS "Setpoint speed increased" (DB390x.DBX2001.2) is enabled.

The minimum gear stage speed is operative only in spindle open loop control mode; the speed of the gear stage can only fall below the minimum limit through:

- Spindle override 0 %
- M5
- S0
- IS "Spindle stop"
- Remove IS "Controller enable"
- IS "Reset"
- IS "Spindle reset"
- IS "Oscillation speed"
- IS "NCSTOP axes and spindle"
- IS "Axis/spindle disable"

13.6.6 Max. encoder limit frequency

WARNING

The maximum encoder limit frequency of the actual spindle position encoder is monitored by the control (the limit can be exceeded). It is the responsibility of the machine tool manufacturer to ensure that the configuration of the spindle motor, gearbox, measuring gearbox, encoder and machine data prevents the maximum speed of the actual spindle position encoder from being exceeded.

Maximum encoder limit frequency exceeded

If the spindle reaches a speed in the open-loop control mode (a high S value has been programmed) which is higher than the max. encoder limit frequency (the max. speed of the encoder may not be exceeded), the synchronization is lost. However, the spindle continues to rotate.

If one of the thread cutting (G33), revolutional feedrate (G95), constant cutting rate (G96, G97) functions is programmed, the spindle speed is reduced automatically so far until the active measuring system works reliably again.

In the "positioning mode" spindle mode and with position-controlled threads (G331, G332) the max. encoder limit frequency is not exceeded.

If the encoder limit frequency is exceeded, the IS "Referenced/synchronized" DB390x.DBX0000.4) is reset for the measurement system and IS "Encoder limit frequency 1 exceeded" (DB390x.DBX0000.2) is enabled.

If the maximum encoder limit frequency has been exceeded and the speed subsequently falls below the encoder frequency in MD36302 ENC_FREQ_LIMIT_LOW (% value of MD36300 ENC_FREQ_LIMIT), the spindle is automatically synchronized with the next zero mark or the next BERO signal.

13.6.7 Target point monitoring

Function

During positioning (the spindle is in "positioning mode"), the system monitors the distance from the spindle (with reference to the actual position) to the programmed spindle set position (target point).

Two limit values can be defined as incremental path (starting at the spindle set position) in the following machine data.

- MD36000 STOP_LIMIT_COARSE (exact stop limit coarse)
- MD36010 STOP_LIMIT_FINE (exact stop limit fine)

Regardless of the two limit values, the positioning of the spindle is always as accurate as the connected spindle measurement encoder, the backlash, the transmission ratio, etc.

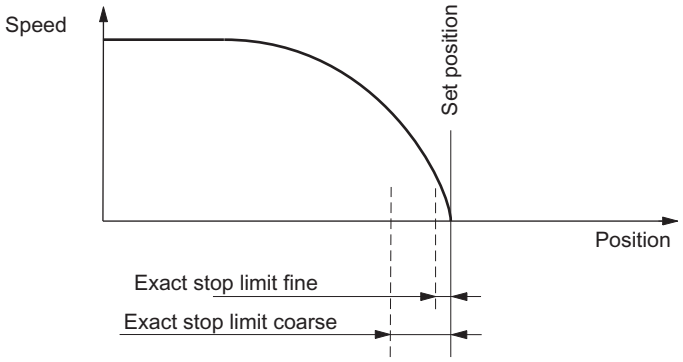


Figure 13-10 Exact stop zones of a spindle for positioning

IS: Position reached with exact stop ...

When the limits MD 36000 and MD 36010 are reached, IS "Position reached with exact stop coarse" (DB390x.DBX0000.6) and IS "Position reached with exact stop fine" (DB390x.DBX0000.7) are output to the PLC.

Block change with SPOS

If the spindle is being positioned with SPOS, the block change will be dependent on the end point monitoring with the IS "Position reached with exact stop fine". All other functions programmed in the block must have achieved their end criterion (e.g. axes ready, all auxiliary functions acknowledged by the PLC).

13.7 Analog spindle

Function

In 808D, an analog spindle is designed for machine running. The spindle is controlled through the rated analog voltage ranging from +10 V to -10 V and two signals in terminals X21-8 and X21-9. The voltage has the corresponding output on the control system.

The analog spindle supports an increment encoder (TTL encoder), which can be connected to the control system directly. You can parameterize the encoder only of an analog spindle. When you set the parameter of the encoder with a step motor shaft, alarm 26006 is thrown out.

Through MD30130 CTRLOUT_TYPE and MD30240 ENC_TYPE, you can switch the rated value output between an analog spindle and an actual spindle. For an analog spindle without any encoder, MD30240 ENC_TYPE[n] must be set to zero.

13.8 Data table

13.8.1 Machine data

Number	Identifier	Name
Channel-specific		
20090	SPIND_DEF_MASTER_SPIND	Master spindle
Axis-specific		
30134	IS_UNIPOLAR_OUTPUT[0]	Setpoint output is unipolar
30300	IS_ROT_AX	Rotary axis
30310	ROT_IS_MODULO	Modulo conversion
30320	DISPLAY_IS_MODULO	Position display
31050 *	DRIVE_AX_RATIO_DENOM[n]	Denominator load gearbox
31060 *	DRIVE_AX_RATIO_NUMERA[n]	Numerator load gearbox
32200 *	POSCTRL_GAIN [n]	Servo gain factor Kv
32810 *	EQUIV_SPEEDCTRL_TIME [n]	Equivalent time constant speed control circuit for feedforward control
34040	REFP_VELO_SEARCH_MARKER	Reference point creep speed
34060	REFP_MAX_MARKER_DIST	Monitoring of zero mark distance
34080	REFP_MOVE_DIST	Reference point distance/destination point for distancecoded system
34090	REFP_MOVE_DIST_CORR	Reference point offset/absolute offset, distancecoded
34100	REFP_SET_POS	Reference point value
34200	ENC_REFP_MODE	Referencing mode
35000	SPIND_ASSIGN_TO_MACHAX	Assignment of spindle to machine axis
35010	GEAR_STEP_CHANGE_ENABLE	Gear stage change possible

Number	Identifier	Name
35040	SPIND_ACTIVE_AFTER_RESET	Spindle active after reset
35100	SPIND_VELO_LIMIT	Maximum spindle speed
35110 *	GEAR_STEP_MAX_VELO[n]	Maximum speed for gear change
35120 *	GEAR_STEP_MIN_VELO[n]	Minimum speed for gear change
35130 *	GEAR_STEP_MAX_VELO_LIMIT[n]	Maximum speed of gear stage
35140 *	GEAR_STEP_MIN_VELO_LIMIT[n]	Minimum speed of gear stage
35150	SPIND_DES_VELO_TOL	Spindle speed tolerance
35200 *	GEAR_STEP_SPEEDCTRL_ACCEL[n]	Acceleration in speed control mode
35210 *	GEAR_STEP_POSCTRL_ACCEL[n]	Acceleration in position control mode
35300	SPIND_POSCTRL_VELO	Position control activation speed
35310	SPIND_POSIT_DELAY_TIME[n]	Positioning delay time
35350	SPIND_POSITIONING_DIR	Positioning direction of rotation for a nonsynchronized spindle
35400	SPIND_OSCILL_DES_VELO	Reciprocating speed
35410	SPIND_OSCILL_ACCEL	Oscillation acceleration
35430	SPIND_OSCILL_START_DIR	Starting direction during oscillation
35440	SPIND_OSCILL_TIME_CW	Oscillation time for M3 direction
35450	SPIND_OSCILL_TIME_CCW	Oscillation time for M4 direction
35500	SPIND_ON_SPEED_AT_IPO_START	Feed enable with spindle in setpoint range
35510	SPIND_STOPPED_AT_IPO_START	Feed enable with stationary spindle
36060	STANDSTILL_VELO_TOL	Threshold velocity "Axis/spindle stationary"
36200 *	AX_VELO_LIMIT [n]	Threshold value for velocity monitoring
36300	ENC_FREQ_LIMIT	Encoder limit frequency
36302	ENC_FREQ_LIMIT_LOW	Encoder limit frequency resynchronization
36720	DRIFT_VALUE	Drift basic value

The machine data marked with * is contained in the parameter set for a gear stage.

13.8.2 Setting data

Number	Identifier	Name
General		
41200	JOG_SPIND_SET_VELO	JOG velocity for the spindle
Spindle-specific		
43230	SPIND_MAX_VELO_LIMS	Programmable spindle speed limit G96

13.8.3 Interface signals

Number	Bit	Name
Axis-specific		
DB30x.DBD0000	-	M function for the spindle (DINT), axis-specific
DB30x.DBD0004	-	S function for the spindle (REAL), axis-specific
DB380x.DBB0000	-	Feed override
DB380x.DBX0001	.7	Override active
DB380x.DBX0001	.5	Position measuring system 1
DB380x.DBX0001	.3	Axis/spindle disable
DB380x.DBX0002	.2	Spindle reset/delete distance-to-go
DB380x.DBX0002	.1	Controller enable
DB380x.DBX2000	.3	Gear changed
DB380x.DBX2000	.0 to .2	Actual gear stage A to ...C
DB380x.DBX2001	.4	Resynchronize spindle during positioning 1 (spindle)
DB380x.DBX2001	.6	Invert M3/M4
DB380x.DBX2002	.7	Set direction of rotation counterclockwise
DB380x.DBX2002	.6	Set direction of rotation clockwise
DB380x.DBX2002	.5	Oscillation speed
DB380x.DBX2002	.4	Oscillation via PLC
DB380x.DBB2003	-	Spindle override
DB390x.DBX0000	.7	Position reached with exact stop fine
DB390x.DBX0000	.6	Position reached with exact stop coarse
DB390x.DBX0000	.4	Referenced/synchronized 1
DB390x.DBX0000	.2	Encoder limit frequency exceeded 1
DB390x.DBX0000	.0	Spindle / no axis
DB390x.DBX0001	.7	Current controller active
DB390x.DBX0001	.6	Speed control loop active
DB390x.DBX0001	.5	Position controller active
DB390x.DBX0001	.4	Axis/spindle stationary ($n < n_{min}$)
DB390x.DBX2000	.3	Change gear stage
DB390x.DBX2000	.0 to .2	Actual gear stage A to ...C
DB390x.DBX2001	.7	Actual direction of rotation clockwise
DB390x.DBX2001	.5	Spindle in setpoint range
DB390x.DBX2001	.2	Setpoint speed increased
DB390x.DBX2001	.1	Setpoint speed limited
DB390x.DBX2001	.0	Speed limit exceeded
DB390x.DBX2002	.7	Active spindle control mode
DB390x.DBX2002	.6	Active spindle mode oscillation mode
DB390x.DBX2002	.5	Active spindle positioning mode
DB390x.DBX2002	.3	Tapping with compensation chuck active
DB390x.DBX2002	.0	Constant cutting rate active (G96)

Feed

14

14.1 Path feedrate F

14.1.1 Path feedrate F

Functionality

The feedrate F is the **path velocity** of the tool along the programmed workpiece contour. The individual axis velocities therefore result from the portion of the axis path in the overall distance to be traversed.

The feedrate F is effective for the interpolation types G1, G2, G3, CIP, and CT and is retained in a program until a new F word is written.

Reference:

Programming and Operating Manual

Dimension units for F: G94, G95

The dimension unit for the F word is determined by G functions:

- G94 F as feedrate in mm/min or inch/min
- G95 F as feedrate in mm/rev of the spindle or inch/rev

  (only meaningful when the spindle is running)

The inch dimension system applies with G700 or system setting "inch" with MD10240 SCALING_SYSTEM_IS_METRIC=0.

Dimension units for F with G96, G97

For **lathes** the group with G94, G95 has been extended by the G96, G97 functions for the **constant cutting rate** (ON/OFF). These functions also influence the S word.

With activated G96 function, the spindle speed is adapted to the currently machined workpiece diameter (transverse axis) such that a programmed cutting rate S remains constant on the tool edge (spindle speed times diameter = constant).

The S word is evaluated as the cutting rate as of the block with G96. G96 is modally effective until cancellation by another G function of the group (G94, G95, G97).

The feedrate F is always evaluated in the unit of dimension of mm/rotation or inch/rotation (as for G95).

Maximum tool path velocity

The maximum path velocity is obtained from the maximum velocities of the relevant axes (MD32000 MAX_AX_VELO) and their proportion of the path. The maximum velocity of an axis stored in the machine data cannot be exceeded.

CFC feedrate override for circles

When machining circular contours using milling tools and the active tool radius compensation (G41/G42), the feedrate at the milling cutter center must be adjusted if the programmed F value is intended to be active at the circular contour. If the **CFC** feedrate override is active, inside and outside circle machining is detected automatically.

The feedrate override can be switched-off using **CFTCP**.

Reference:

Programming and Operating Manual

Interface signals

If the revolutional feedrate is active, IS "Revolutional feedrate" (DB3300.DBX0001.2) is set.

If the G96/G332 function is active, the IS "Constant cutting rate active" (DB390x.DBX2002.0) is set for the spindle.

Alarms

- If no F word is programmed at G1, G2, G3, ..., alarm 10860 is issued. An axis movement is not possible. However, please note: SD42110 DEFAULT_FEED!
- If F0 is programmed, alarm 14800 is issued.
- If G95 is active and the spindle is stationary, an axis movement is not possible. No alarm is issued.

Notes

- If the "Dry run feedrate" function is activated and the program is started, the feedrates programmed in combination with G1, G2, G3, CIP, CT will be replaced by the feedrate value stored in SD42100 DRY_RUN_FEED, see Section "Program processing with dry run feedrate (DRY) (Page 112)".
- The velocity of the traversing movement of an axis in the JOG mode is determined by the machine data/setting data.

14.1.2 Feedrate with G33, G34, G35 (thread cutting)

Types of thread cutting

G33 - thread with constant pitch

G34 - thread with (linearly) increasing pitch

G35 - thread with (linearly) decreasing pitch

Axis velocity

With respect to G33, G34, or G35 threads, the axis velocity for the thread length results from the set spindle speed and the programmed pitch. However, the maximum axis velocity defined in MD32000 MAX_AX_VELO cannot be exceeded.

The feedrate F is not relevant. It is, however, kept in the memory.

The axis velocity, e.g. for a cylinder thread, results from the set spindle speed (S) and programmed pitch (K):

Fz [mm/min] = speed S [rev/min] * pitch K [mm/rev]

Note

For G34 and G35 the pitch change in mm/rev^2 is programmed under the F address.

Reference:

Programming and Operating Manual

NC stop, single block

NC stop and single block are only active after completion of thread chaining.

Information

- The spindle speed override switch must remain unchanged during thread machining (tapping).
- The feedrate override switch is irrelevant in a block with G33, G34, G35.

Programmable runin and runout path: DITS, DITE

The run-in and run-out path is to be traversed in addition to the required thread. The starting and braking of the axis (both axes in case of a tapered thread) are performed in these areas. This path depends on the pitch, spindle speed, and the axis dynamics (configuration).

If the available path for run-in or run-out is limited, it may be necessary to reduce the spindle speed so that this path is sufficient. In this case, the run-in and run-out paths can be specified separately in the program to achieve favorable cutting values and short machining times or to simplify the handling of this issue.

If no values are specified, the values from the setting data (SD) apply. The specifications in the program are written in SD42010 THREAD_RAMP_DISP[0] ... [1].

If this path is not sufficient for traversing at the configured axis acceleration, the axis is overloaded in terms of acceleration. Alarm 22280 ("Programmed run-in path too short") is then issued for the thread run-in. The alarm is purely for information and has no effect on part program execution.

The run-out path acts as an approximate distance at the end of the thread. This achieves a smooth change in the axis movement when retracting.

Programming

DITS= ...: Run-in path of the thread

DITE= ...: Run-out path of the thread

Reference:

Programming and Operating Manual

SD42010

Only paths, and not positions, are programmed with DITS and DITE.

With the part program instructions, the setting data SD42010 THREAD_RAMP_DISP[0], ...[1] defines the following acceleration response of the axis during thread cutting ([0]-run-in, [1]-run-out):

- SD42010 = < 0 to -1:

  Starting/braking of the feedrate axis at configured acceleration rate. Jerk according to current BRISK/SOFT programming.

- SD42010 = 0:

  Abrupt starting/braking of the feedrate axis on thread cutting.

- SD42010 = > 0:

  The thread run-up/deceleration distance is specified. To avoid technology alarm 22280, the acceleration limits of the axis must be observed in case of very small run-in and run-out paths.

Note

DITE acts at the end of the thread as an approximate distance. This achieves a smooth change in the axis movement.

Pitch change F with G34, G35

If you already know the starting and final lead of a thread, you can calculate the pitch change F to be programmed according to the following equation:

$$F = \frac{|K_e^2 - K_a^2|}{2 * L_G} \text{ [mm/rev}^2\text{]}$$

The identifiers have the following meanings:

K_e Pitch of axis target point coordinate [mm/rev]

K_a Initial pitch (progr. under I and K) [mm/rev]

L_G Thread length in [mm]

14.1.3 Feedrate for G63 (tapping with compensation chuck)

Feedrate F

In the case of G63 it is necessary to program a feedrate F. It must be suitable for the selected spindle speed S (programmed or set) and for the pitch of the drill:

Feedrate F [mm/min] = speed S [rev/min] x pitch [mm/rev]

The compensation chuck absorbs possible path differences of the drill axis to a limited extent.

Reference:

Programming and Operating Manual

14.1.4 Feedrate for G331, G332 (tapping without compensation chuck)

Axis velocity

With respect to G331/G332 tapping, the axis velocity for the thread length results from the effective spindle speed S and the programmed pitch. However, the maximum axis velocity defined in MD32000 MAX_AX_VELO cannot be exceeded.

The feedrate F is not relevant. It is, however, kept in the memory.

Interface signal

If the G331/G332 function is active, the IS "Tapping without compensation chuck active" (DB390x.DBX2002.3) is set for the spindle.

Note

The tapping may only be carried out without a compensation chuck if an exact dynamic adjustment of the spindle and the relevant axis has been performed. With G331/G332 the parameter set n (0...5) of the axis becomes effective automatically. This parameter set also applies to the current gear stage of the spindle (M40, M41 to M45 - see also Chapter "Spindle (Page 163)").

In general, the axis is adjusted to the slower spindle.

Reference:

Programming and Operating Manual

14.1.5 Feedrate for chamfer/rounding: FRC, FRCM

Chamfer/rounding

You can insert the chamfer (CHF or CHR) or rounding (RND) elements into a contour corner. If you wish to round several contour corners sequentially by the same method, use "Modal rounding" (RNDM).

You can program the feedrate for the chamfer/rounding with FRC=... (non-modal) or FRCM= ... (modal). If FRC/FRCM is not programmed, the normal feedrate F is applied.

Programming

FRC=...	Non-modal feedrate for chamfer/rounding	
	Value > 0:	Feedrate in mm/min (G94) or mm/rev. (G95)
FRCM=...	Modal feedrate for chamfer/rounding	
	Value > 0:	Feedrate in mm/min (G94) or mm/rev. (G95)
		Modal feedrate for chamfer/rounding ON
	Value = 0:	Modal feedrate for chamfer/rounding OFF
		Feedrate F applies to the chamfer/rounding

Notes

- F, FRC, FRCM are not active when a chamfer is traversed with G0. If the feedrate F is active for chamfer/rounding, it is by default the value from the block which leads away from the corner. Other settings can be configured via machine data MD20201 CHFRND_MODE_MASK.
- A maximum of three blocks without corresponding information may be put between two blocks containing traversing information for chamfer/rounding (axes of the plane). In the case of more blocks without axis information in the plane and existing instructions for inserting chamfer or rounding, an alarm is triggered.

14.2 Rapid traverse G0

Application

The rapid traverse movement G0 is used for rapid positioning of the tool, but not for direct workpiece machining. All axes can be traversed simultaneously. This results in a straight path.

For each axis, the maximum speed (rapid traverse) is defined in machine data MD32000 MAX_AX_VELO. If only one axis traverses, it uses its rapid traverse. If, for example, two axes are traversed simultaneously, the path velocity (resulting velocity) is selected to achieve the maximum possible path velocity under consideration of both axes.

If, for example, two axes have the same maximum velocity and also travel the same path, the path velocity = 1.41 * max. axis velocity.

The feedrate F is not relevant for G0. It is, however, kept in the memory.

Rapid traverse override

In the "AUTO" operating mode, it can be set through the "Machine" operation area -> "Prog. cont." softkey that the feedrate override switch also applies to the rapid traverse. The active function is displayed with ROV in the status line. HMI to PLC sets the IS "Feedrate override for rapid traverse selected" (DB1700.DBX0001.3). The PLC user program must place this signal on the IS "Rapid traverse override active" (DB3200.DBX0006.6).

14.3 Feedrate control

14.3.1 Overview

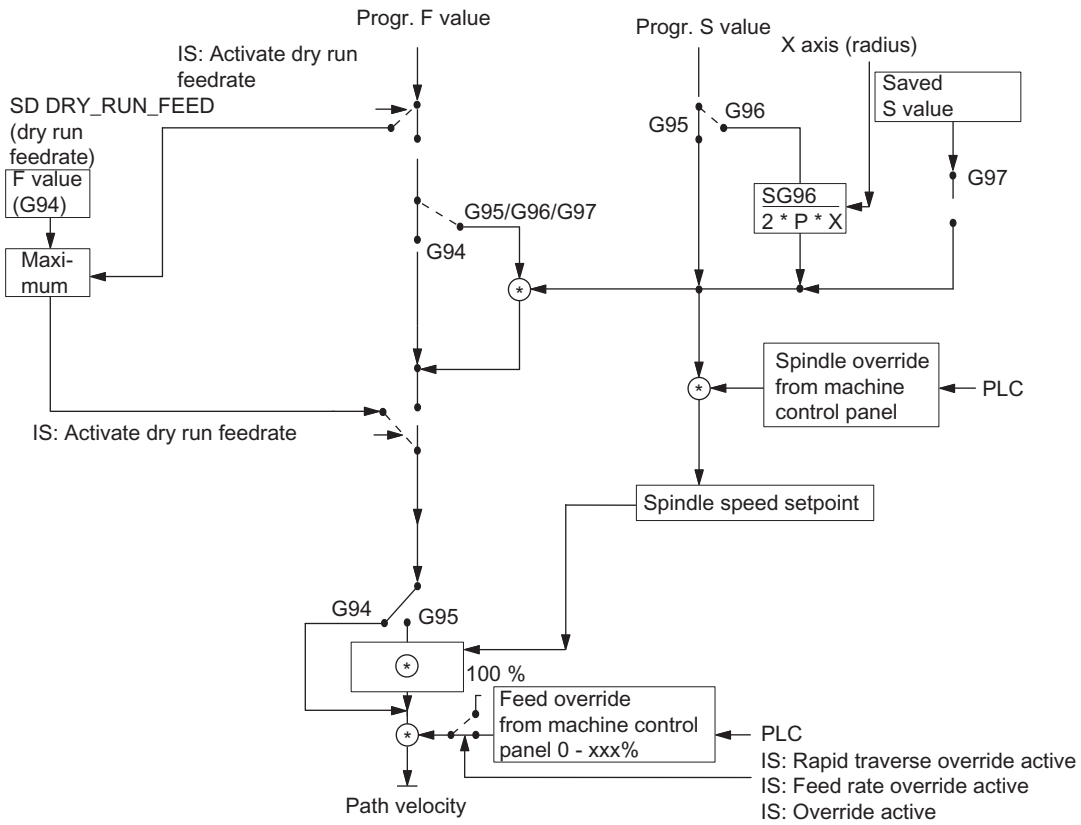


Figure 14-1 Possibilities for programming and controlling the feedrate

14.3.2 Feedrate disable and feedrate/spindle stop

General

The "Feed disable" or "Feed/spindle stop" brings the axes to a standstill. The path contour is maintained (exception: G33 block).

Feed disable

The channel-specific interface signal "Feed disable" (DB3200.DBX0006.0) will stop all axes (geometry and special axes) in all operating modes.

This feed disable is **not** effective if G33 is active; it is, however, active with G63, G331, G332.

Feed stop for axes in the WCS

The "Feed stop" interface signals (DB3200.DBX1000.3, DB3200.DBX1004.3, and DB3200.DBX1008.3) are used to stop the geometry axes (axes in the WCS) during traversing in the workpiece coordinate system (WCS) in the JOG mode.

Axis-specific feed stop

The axis-specific "Feed stop" interface signal (DB380x.DBX0004.3) is used to stop the relevant machine axis.

In the "AUTO" mode: If the "Feed stop" is performed for a path axis, all the axes traversed in the current block and all axes participating in the axis group are stopped.

Only the current axis is stopped in JOG mode.

The axis specific "Feed stop" is active when G33 is active (but: contour deviations = thread error!).

Spindle stop

The "Spindle stop" interface signal (DB380x.DBX0004.3) is used to stop the spindle.

"Spindle stop" is active with G33 and G63.

Note

Contour deviations = thread error!

14.3.3 Feedrate override via a machine control panel

General

The operator can use the feedrate override switch to increase or decrease the path feedrate relative to the programmed feedrate in percent with immediate effect. The feedrates are multiplied by the override values.

An override between 0 and 120% can be programmed for the path feedrate F.

The rapid traverse override switch is used to reduce the traversing velocity when testing a part program.

An override between 0 and 100% can be programmed for the rapid traverse.

The spindle override can be used to modify the spindle speed and the cutting rate (with G96). The override can be between 50 and 120%.

The override is not permitted to exceed the machine specific acceleration and speed limits or generate a contour error.

The override acts on the **programmed values** before limits intervene.

Channel-specific feedrate and rapid traverse override

One enable signal and one byte are provided on the PLC interface for the override factor in percent for feedrate and rapid traverse:

- IS "Feedrate override" (DB3200.DBB0004)
- IS "Feedrate override active" (DB3200.DBX0006.7)
- IS "Rapid traverse override" (DB3200.DBB0005)
- IS "Rapid traverse override active" (DB3200.DBX0006.6)

The interface for the override (value) is supplied by a machine control panel via the PLC to the NC and it is Gray-coded.

An active feedrate override acts on all path axes. An active rapid traverse override acts on all axes traversing with rapid traverse.

If there is no dedicated rapid traverse override switch, the feedrate override switch can be used. In this case, feedrate overrides above 100% are limited to 100% for rapid traverse override.

The override to be active can be selected via the PLC or operator panel.

If the selection is made using the operator panel (display: ROV), the IS "Feedrate override for rapid traverse selected" (DB1700.DBX0001.3) is set and must be transferred by the PLC user program to the IS "Rapid traverse override active" (DB3200.DBX0006.6). The value itself is to be transferred by the PLC user program from a machine control panel to the IS "Rapid traverse override" (DB3200.DBB0005).

The channel-specific feedrate and rapid traverse overrides are inactive if G33, G63, G331 and G332 are active.

Axis-specific feedrate override

One enable signal and one byte for the feedrate override factor in percent are available on the PLC interface for each axis:

- IS "Feedrate override" (DB380x.DBB0000)
- IS "Override active" (DB380x.DBX0001.7)

If G33, G331, G332, G63 are active, the axis-specific feedrate override has no effect (is internally set to a fixed value of 100%).

Spindle override

One enable signal and one byte for the spindle override factor in percent are available on the PLC interface for each spindle:

- IS "Spindle override" (DB380x.DBB2003)
- IS "Override active" (DB380x.DBX0001.7)

The additional signal IS "Feedrate override for spindle valid" (DB380x.DBX2001.0) allows the PLC user program to determine that the value of the IS "Feedrate override" (DB380x.DBB0000) should apply.

The spindle override is active with G33, but it should not be actuated for reasons of accuracy; also active with G331, G332. In the case of G63, the spindle override is set to a fixed value of 100%.

Override active

The set override values are effective in all operating modes and machine functions. This applies if the IS "Rapid traverse override active", "Feedrate override active" or "Override active" are set.

An override factor of 0% acts as a feedrate disable.

Override inactive

When the override is inactive (i.e. the above interface signals are set to "0"), the override factor "1" is used internally for all switch positions (except from the 1st position), i.e. the override is **100%**.

Note

The 1st switch position of the Gray-coded interfaces for the value represents a special case. In this case, the override factor of the 1st switch position is also used if the IS "Rapid traverse override active", "Feedrate override active", "Override active" are not set. Thus **0%** is issued as the override value for axes (acts the same as "Feed disable"). The following applies to the spindle if the IS "Override active" is not set: Override value **50%**.

14.4 Data table

14.4.1 Machine/setting data

Number	Identifier	Name
General machine data		
10240	SCALING_SYSTEM_IS_METRIC	Basic system metric
Channel-specific machine data		
20201	CHFRND_MODE_MASK	Specifications regarding the chamfer/rounding behavior
Axis-specific machine data		
32000	MAX_AX_VELO	Maximum axis velocity
35100	SPIND_VELO_LIMIT	Maximum spindle speed
Channel-specific setting data		
42100	DRY_RUN_FEED	Dry run feedrate
42010	THREAD_RAMP_DISP	Acceleration behavior of the feedrate axis when thread cutting
42110	DEFAULT_FEED	Default value for path feed

14.4.2 Interface signals

Number	Bit	Name
Channel-specific		
DB3200.DBX0000	.6	Activate dry run feed
DB3200.DBX0004	-	Feed override
DB3200.DBX0005		Rapid traverse override
DB3200.DBX0006	.0	Feed disable
DB3200.DBX0006	.6	Rapid traverse override active
DB3200.DBX0006	.7	Feed rate override active
DB3200.DBX1000	.3	Feed stop, geometry axis 1
DB3200.DBX1004	.3	Feed stop, geometry axis 2
DB3200.DBX1008	.3	Feed stop, geometry axis 3
DB1700.DBX0000	.6	Dry run feed rate selected
DB1700.DBX0001	.3	Feed rate override selected for rapid traverse
DB3300.DBX0001	.2	Revolutional feed rate active
Axis/spindle-specific		
DB380x.DBB0000	-	Feed override
DB380x.DBB2003	-	Spindle override
DB380x.DBX0001	.7	Override active (axis or spindle)
DB380x.DBX2001	.0	Feedrate override for spindle valid
DB380x.DBX0004	.3	Feed stop/spindle stop
DB390x.DBX2002	.0	Constant cutting rate active (spindle)
DB390x.DBX2002	.3	Tapping without compensation chuck active (spindle)

Tool: Tool Compensation 15

15.1 Tool and tool compensation overview

Characteristics

The SINUMERIK 808D control is capable of calculating the tool compensation data for different tool types (drill, milling cutter, turning tool, ...).

- Length compensation
- Radius compensation
- Storage of the tool compensation data in the tool offset memory
  - Tool identification with T numbers from 0 to 32000
  - Definition of a tool with a maximum of nine cutting edges (offset blocks) through D number
  - Cutting edge is described by tool parameters:

    Tool type

    Geometry: Length/radius

    Wear: Length/radius

    Cutting edge position (for turning tools)
- Tool change selectable: Immediately with T command or through M6
- Tool radius compensation
  - Compensation active for all interpolation types: linear and circular
  - Compensation at outer corners selectable: transition circle (G450) or equidistant intersection (G451)
  - Automatic detection of outer/inner corners

Detailed description:

References:

Programming and Operating Manual

15.2 Tool

Select a tool

A tool is selected in the program with the T function. Whether the new tool is immediately loaded with the T function or with M6 depends on the setting in MD22550 TOOL_CHANGE_MODE (new tool offset with M function).

Value range of T

The T function can assume integer values from T0 (no tool) to T32000 (tool with the number 32000).

Up to 64 tools can be stored in the control system simultaneously.

15.3 Tool offset

Tool compensation through D function

A tool can have up to nine cutting edges. The nine tool cutting edges are assigned to the D functions D1 to D9.

Up to 128 data fields (D numbers) for tool compensation blocks can be stored in the control system simultaneously.

The tool cutting edge is programmed with D1 (edge 1) to D9 (edge 9). The tool cutting edge always refers to the currently active tool. An active tool cutting edge (D1 to D9) without an active tool (T0) is inactive. Tool cutting edge D0 deselects all tool offsets of the active tool.

Selection of the cutting edge when changing tool

When a new tool (new T number) has been programmed and the old one replaced, the following options are available for selecting the cutting edge:

- The cutting edge number is programmed
- The cutting edge number is not programmed D1 is active automatically.

Activating the tool offset

D1 to D9 activates the tool compensation (offset) for a cutting edge on the active tool. Tool length compensation and tool radius compensation can be activated at different times:

- Tool length compensation (TLC) is performed on the first traversing motion of the axis on which the TLC is to act. This traversing motion must be a linear interpolation (G0, G1).
- Tool radius compensation (TRC) becomes active when G41/G42 is programmed in the active plane (G17, G18 or G19). The selection of tool radius compensation with G41/G42 is only permitted in a program block with G0 (rapid traverse) or G1 (linear interpolation).

Detailed description of the tool compensation (offset) including tool radius compensation:

Reference:

Programming and Operating Manual

15.4 Special handling of tool compensation

For SINUMERIK 808D, tool compensation (offset) can be handled as follows.

Influence of setting data

Using specific setting data the operator / programmer can influence the calculation of the length compensation of the used tool:

- SD42940 TOOL_LENGTH_CONST
  (allocation of the tool length components to the geometry axes)
- SD42950 TOOL_LENGTH_TYPE
  (allocation of the tool length components independent of tool type)

Note

The modified setting data will become effective with the next cutting edge selection.

Tool length and plane change (SD42940 TOOL_LENGTH_CONST)

Value of the setting data equal to 0:

The behavior corresponds to the standard definition: The lengths 1 to 3 in geometry and wear are assigned to the 1st to 3rd axes of the plane according to the active G17 to G19 and according to the tool type. If the active G17 to G19 changes, the axis assignment for the lengths 1 to 3 also changes because abscissa, ordinate and application are allocated to different geometry axes.

Reference:

Programming and Operating Manual

Value of the setting data not equal to 0:

The assignment of the tool lengths 1 to 3 in geometry and wear to the geometry axes are performed according to the SD value and are **not** changed if the machining plane (G17 to G19) changes.

The assignment of the tool lengths 1 to 3 to the geometry axes for **turning tools** (tool types 500 to 599) results from the value of the setting data SD42940 in accordance with the following table:

Plane/value	Length 1	Length 2	Length 3
17	Y	X	Z
18*)	X	Z	Y
19	Z	Y	X
-17	X	Y	Z
-18	Z	X	Y
-19	Y	Z	X

*) Each value not equal to 0 which is not equal to one of the six listed values is evaluated as the value for 18.

With respect to the values with a negative sign the assignment of length 3 is identical, length 1 and 2 are exchanged - compared to the assignment with the corresponding positive values.

The following table shows the assignment of the tool lengths 1 to 3 to the geometry axes for **drills / milling cutters** (tool types 100 to 299):

Plane/value	Length 1	Length 2	Length 3
17*)	Z	Y	X
18	Y	X	Z
19	X	Z	Y
-17	Z	X	Y
-18	Y	Z	X
-19	X	Y	Z

*) Each value not equal to 0 which is not equal to one of the six listed values is evaluated as the value for 17.

With respect to the values with a negative sign the assignment of length 1 is identical, length 2 and 3 are exchanged - compared to the assignment with the corresponding positive values.

Note

For representation in tables, it is assumed that geometry axes 1 to 3 are named X, Y, Z. The axis order (1st, 2nd and 3rd geometry axis) but not the axis identifier determines the assignment between an offset and an axis.

Length compensation for tool type (SD42950 TOOL_LENGTH_TYPE)

Value of the setting data equal to 0:

The behavior corresponds to the standard definition: The lengths 1 to 3 in geometry and wear are assigned to the actual **tool type** (milling cutter / drill or turning tool).

Reference:

Programming and Operating Manual

Value of the setting data not equal to 0:

The assignment of the tool lengths is always independent of the actual tool type.
Value 1: Length assignment always as for milling tools.
Value 2: Length assignment always as for turning tools.

Notes

- The influence of these two setting data only refers to tool lengths. The tool radius is not affected.
- If SD42940 TOOL_LENGTH_CONST is set not equal to 0 and the value in SD42950 TOOL_LENGTH_TYPE is 1 or 2, the related table for the assigned tool type (milling or turning tool) applies in SD42940.

Example

SD42940 TOOL_LENGTH_CONST =18
SD42950 TOOL_LENGTH_TYPE =2

Explanation:

The active tool with the active D number always behaves as a turning tool in the length compensation (-> SD42950 =2).

The length assignment is performed in all planes G17 to G19 as for G18 (-> SD42940=18):
Length 1 -> X axis
Length 2 -> Z axis
if Y axis exists: Length 3 -> Y axis

The tool radius acts according to the actual tool type and the active plane.

15.5 Data table

15.5.1 Machine data

Number	Identifier	Name
Channel-specific		
22360	TOOL_PARAMETER_DEF_MASK	Definition of tool parameters
22550	TOOL_CHANGE_MODE	New tool offsets with M function

15.5.2 Interface signals

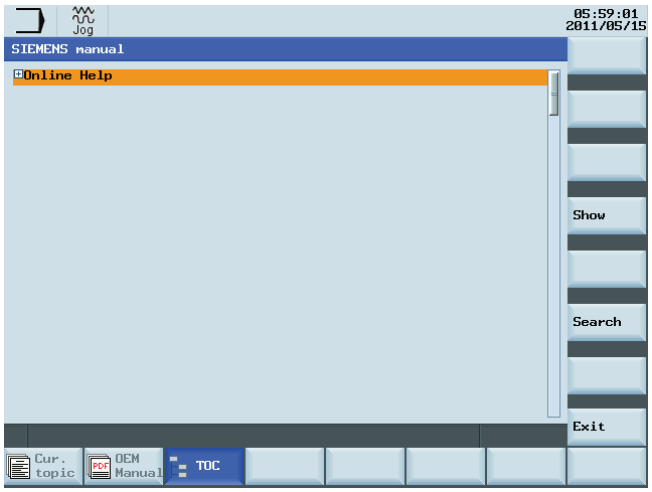
Number	Bit	Name
Channel-specific		
DB2500.DBX0008	.0	T function 1 change
DB2500.DBX0010	.0	D function 1 change
DB2500.DBD2000	-	T function 1
DB2500.DBD5000	-	D function 1
DB2500.DBX1000	.6	M6
DB3200.DBX0013	.5	Deactivate workpiece counter

Special functions

16

16.1 Calling an online help

A Siemens online help is available for your reference. You can press the <HELP> key to call it:



Press the <INPUT> key on the PPU or the "Show" softkey to display it, or press "Exit" to exit the screen of the online help.

OEM online help

You can also create your own online help in text files, and upload the help into the control system using a USB stick.

To create your own online help, you must use the existing help file format. For example:

```
#{XE"Chapter 2"}                                          > create a bookmark that will be displayed in help content list.
#{HL1}                                                    > define a headline with depth 1.
Chapter 2

#{HL2}                                                    > define a headline with depth 2.
How to find the Online Help?

You can find the Online Help by pressing the HELP key:    > text that follows the headline.
#{BITMAP"Images/1.bmp"}                                   > insert a bitmap 1 in the text from the folder "Images".

#{HL2}
How to open the Online Help?

1. Press the right arrow key, and you can view the Online Help you have
created.
2. Press the INPUT key or softkey "Show" to show it.

#{XREF"Chapter 1"}{text 1}{Go to Chapter 1}               > create a hyperlink to Chapter 1.
```

Note

You must end the text by pressing the Enter key; otherwise the online help does not work properly.

The table below gives the detailed information about the commands you can use in your help texts:

Command	Description
#{XE "**BookmarkName**"}	Will create a bookmark named **BookmarkName**. The command must be followed by an HL command, which will be used as description in the help index. These bookmarks will be displayed in the help content list.
#{HL**{depth}**} {**depth**} = 1 - 5	Defines a headline. The parameter {depth} defines the headline depth.
#{NPAGE}	Starts a new help page
#{BOOKMARK "BookmarkName"}	Sets a hidden bookmark named **BookmarkName**, which occurs in the help index. It can be used in the **XREF** command to create a hyperlink.
#{XREF "BookmarkName"}{file name}{Display text of hyperlink}	Will create a hyperlink in the help text. The destination **BookmarkName** can be a bookmark created via **BOOKMARK** or **XE** command.
#{BITMAP "no_ref.bmp"}	Inserts a bitmap in the text.
#{SCOLOR **{color}**} {color} = RED, ORANGE, BLACK, BLUE, GREEN, YELLOW, WHITE	Changes the color of the following text to the specified one

Uploading an OEM online help using a USB stick

To upload an OEM online help using a USB stick, proceed as follows:

1. Create your own file(s) for an online help and save the file(s) in the USB stick.

   The possible file formats are **.txt**, **.png**, and **.bmp**. Because the OEM online help supports multiple languages, you need to first create folders for different languages. You can create, for example, the following folder structure, in the USB stick.

   First level:

   ```
   ..
   chs                    DIR          11/10/11  00:19:18
   eng                    DIR          11/10/11  00:19:18
   ptb                    DIR          11/10/11  00:19:18
   rus                    DIR          11/10/11  00:19:18
   ```

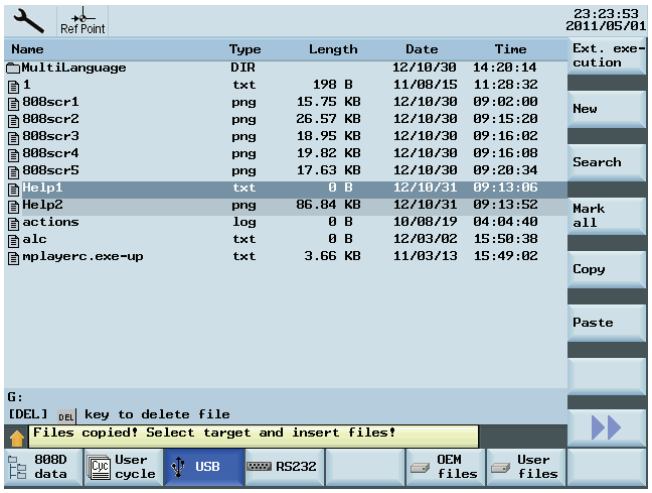
   Second level:

   ```
   ..
   milling                DIR          11/10/11  00:19:18
   turning                DIR          11/10/11  00:19:18
   ```

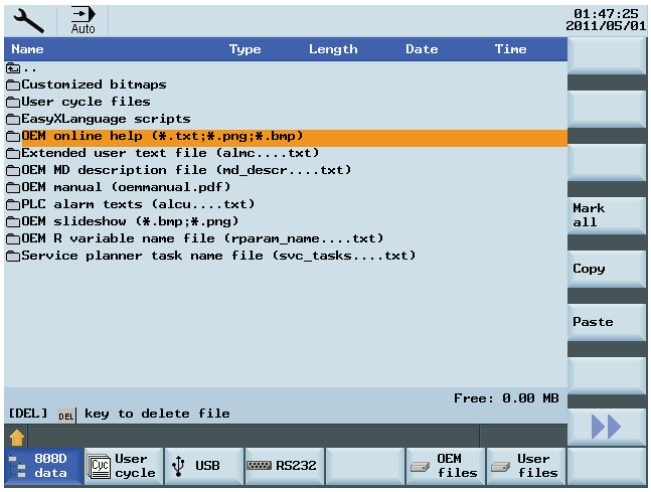
   Third level:

   ```
   ..
   manual                 DIR          11/10/11  00:20:33
   ```

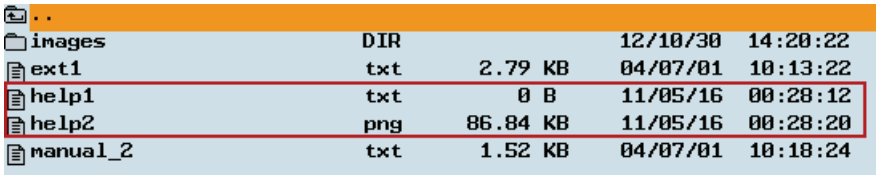
2. Copy the four first-level folders to the "OEM online help (*.txt; *.png; *.bmp)" folder. For how to find this folder, see the subsequent steps.
3. Insert the USB stick into the USB interface in the front of the PPU.
4. In the "SYSTEM" operating area, press the "Sys. data" softkey.
5. Press the "USB" softkey.
6. Select the online help file(s) using <SELECT>, and then press the "Copy" softkey.



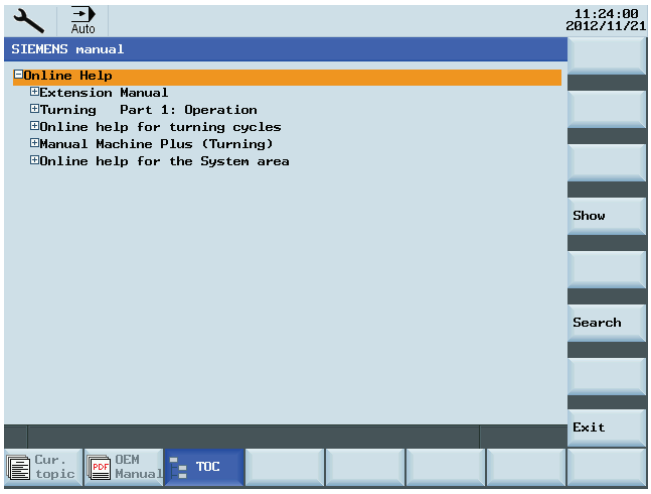
7. Press the "808D data" softkey, and press <INPUT> to access the "HMI data" folder. Then select the "OEM online help (*.txt; *.png; *.bmp)" folder by using the Up and Down arrow keys.



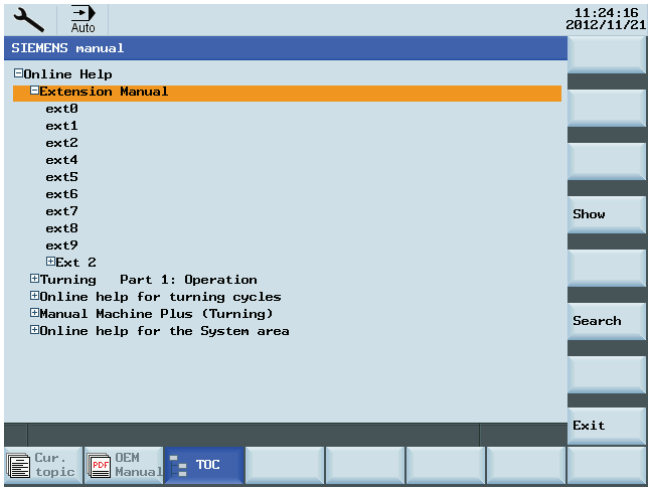
8. Press <INPUT> to access the "OEM online help (*.txt; *.png; *.bmp)" folder, and then enter the "manual" folder, that is, the above-mentioned third-level folder. Press the "Paste" softkey to paste the copied file(s) under this folder.



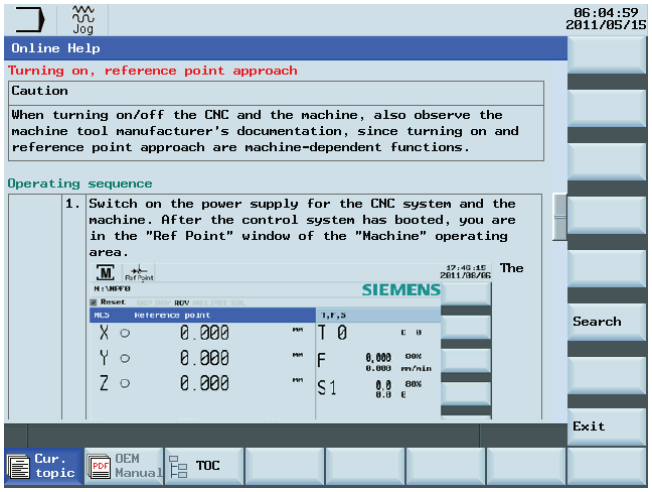
9. Press <HELP> and press the right arrow key. Then you can view your own online help, as shown in following example.



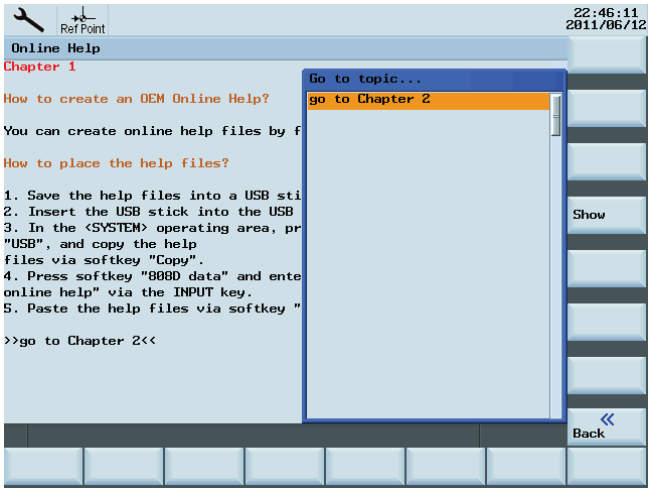
You can view the help content list by pressing the right arrow key.



10. Directly press <INPUT> or the "Show" softkey to show your online help, or select one help content and then view it using <INPUT> or the "Show" softkey.



11. If you have created hyperlinks in the online help text, use the "Go to topic" softkey and then press the "Show" softkey to go to the linked target.



12. Press "Exit" to exit the online help or press the "TOC" softkey to return to the online help main menu.

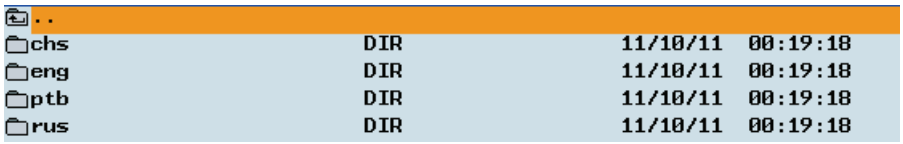
Uploading an OEM manual using a USB stick

To upload an OEM manual using a USB stick, proceed as follows:

1. Create your own file(s) for an OEM manual and save the file(s) in the USB stick.

   The file format must be **oemmanual.pdf**. Because the OEM online help supports multiple languages, you need to first create folders for different languages. You can reate, for example, the following folder structure, in the USB stick.

   First level:

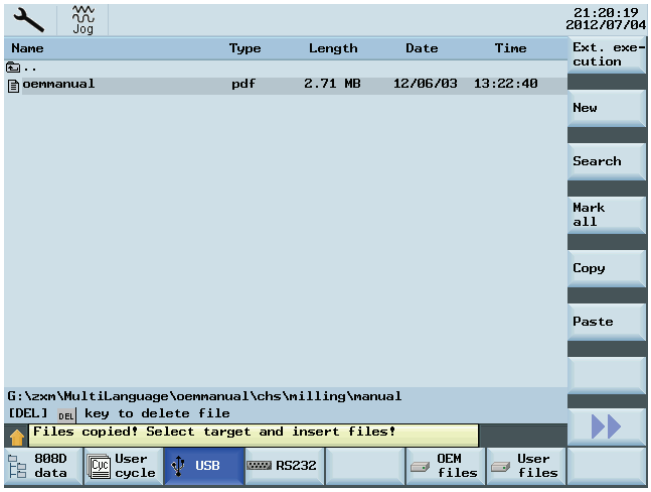
   Second level:

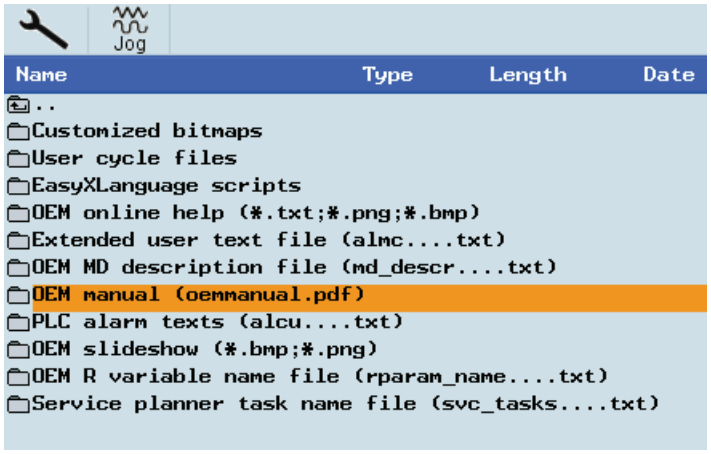
   Third level:

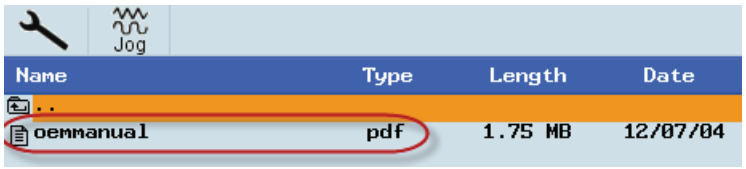
2. Copy the four first-level folders to the "OEM manual (oemmanual.pdf)" folder. For how to find this folder, see the subsequent steps.
3. Insert the USB stick into the USB interface in the front of the PPU.
4. In the "SYSTEM" operating area, press the "Sys. data" softkey.
5. Press the "USB" softkey.
6. Select the OEM manual file(s) using <INPUT>, and then press the "Copy" softkey.



7. Press the "808D data" softkey, and press <INPUT> to access the "HMI data" folder. Then select the "OEM manual (oemmanual.pdf)" folder by using the Up and Down arrow keys.



8. Press <INPUT> to access the "OEM manual (oemmanual.pdf)" folder, and then enter the "manual" folder, that is, the above-mentioned third-level folder. Press the "Paste" softkey to paste the copied file(s) under this folder.



9. Press <HELP> and then press the "OEM Manual" softkey. Then you can view your own OEM manual.
10. Press "Exit" to exit the OEM manual.

NOTICE
Poor performance of the system Do not upload one or more OEM files of too large size; otherwise, the system performance will be reduced.

16.2 Calling a standard cycle with auxiliary functions

For the SINUMERIK 808D control system, you can call user cycles with M codes or T codes. With this function, you can perform operations such as changing machine tools.

Note

M codes or T codes for calling user cycles **must not** be in the same program segment.

Calling cycles with "M6"

Configure the parameters shown in the table below to activate an M code for calling a standard cycle:

No.	Name	Unit	Value	Description
22550	TOOL_CHANGE_MODE	-	1	Activating tool parameters with an M code
22560	TOOL_CHANGE_M_CODE	-	206	The M code for activating tool parameters
10715	M_NO_FCT_CYCLE[0]	-	6	Calling the standard cycle with M06
10716	M_NO_FCT_CYCLE_NAME[0]	-	"TOOL"	Name of the standard cycle

For the format of a standard cycle, refer to the example shown below:

```
%_N_TOOL_SPF
;$PATH=/_N_CUS_DIR
PROC TOOL SAVE DISPLOF
IF $P_ISTEST GOTOF _END
IF $P_SEARCH<>0 GOTOF _END
IF $P_TOOLNO==$P_TOOLP GOTOF _NO
G500 D0
G75 Z=0
SPOS=$MN_USER_DATA_FLOAT[0]
MSG(“Ready to change tool*** Original tool number: T”<<$P_TOOLNO)
M206
STOPRE
G153 G01 Z0 F2000 ;G153
MSG(“Ready to change tool *** Original tool number: T”<<$P_TOOLP)
GOTOF _END
_NO:
MSG(“No action *** Reason: programming tool number = spindle tool number”)
_END:
M17
```

Calling cycles using the "T" function

Configure the parameters shown in below table to activate a T code for calling a standard cycle:

No.	Name	Unit	Value	Description
22550	TOOL_CHANGE_MODE	-	0	Activating tool parameters with an M code
10717	T_NO_FCT_CYCLE[0]	-	"TOOL"	Calling the standard cycle with M06

The format of the standard cycle is the same with that of M codes. The tool number for programming will be saved into system variable $C_T.

Descriptions of frequently used system variables

Variables	Descriptions
$P_ISTEST	Program testing status; boolean variable
$P_SEARCH	Program searching status; boolean variable
$P_SEARCHL	Program searching status; real numbers: 1-, 2-, 3-
$P_TOOLNO	Tool number in the spindle turret
$P_TOOLP	Programming tool number
$C_T	Programming tool number. $P_TOOLP is inactive when the program code T calls a tool changing cycle that is defined with MD10717. The tool number is then represented with "$C_T".
$TC_DP1[Tool number, 1]	Tool type
$TC_DP3[Tool number, 1]	Tool's geometrical parameter: tool length 1
$TC_DP6[Tool number, 1]	Tool's geometrical parameter: tool radius
$TC_DP12[Tool number, 1]	Tool wear: the direction of length 1
$TC_DP15[Tool number, 1]	Tool wear: the direction of radius
$TC_DP24[Tool number, 1]	Tool's dimension: 0: normal 1: oversize
$TC_DP25[Tool number, 1]	Number of the tool turret
_TM[n]	Global user data (integral)
_ZSFR[n]	Global user data (float) **NOTE:** Since this data has been used in the Siemens standard technology cycles, ensure that there is no conflict with the technology cycles when you are using this data.

16.3 Display function

Displaying the part timer

The part timer is available for the SINUMERIK 808D to count the following time periods:

Time	Corresponding system variable	Description
Run time	$AC_OPERATING_TIME	Total time for running programs in **AUTO** mode
Cycle time	$AC_CYCLE_TIME	Run time of a selected program
Cutting time	$AC_CUTTING_TIME	Cutting time (G01, G02, G03) of a selected program
Setup time [1)]	$AN_SETUP_TIME	Time elapsed since the last power-on with default values
Power on time [1)]	$AN_POWERON_TIME	Time elapsed since the last normal power-on
Remain time [2)]	-	Remaining time for running the current program.

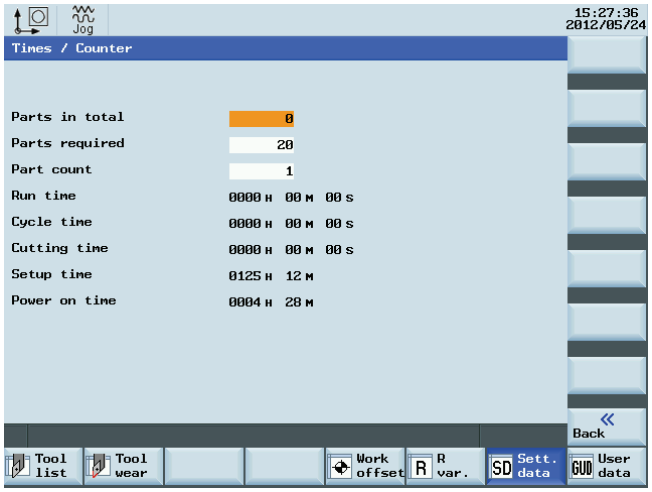
1) The remaining time has no corresponding system variable, and can be counted only after a cycle of a part program has successfully run.

2) Both the setup time and the power on time are counted automatically after the controller has been powered on.

By default, the run time, the cycle time, the setup time and the power on time are displayed. The cutting time can only be counted after being activated with MD27860:

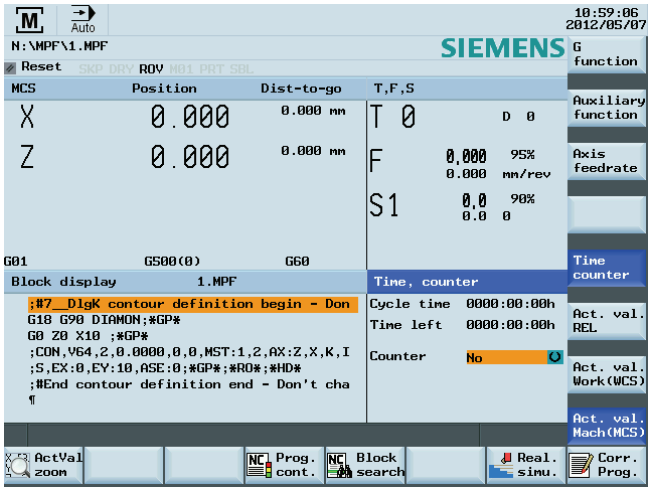
No.	Name	Value	Descriptions
27860	PROCESSTIMER_MODE	Actual value	Activation of counting for following program runtime: • Run time • Cycle time • Cutting time

In the "OFFSET" operating area, press the horizontal softkey "Sett. data" > Vertical softkey "Time counter" and you can open the "Times / Counter" window:



Under the "AUTO" mode in the "MACHINE" operating area, the time counter (vertical softkey "Time counter") can also be displayed counting following time:

- Cycle time
- Time left



Displaying the part counter

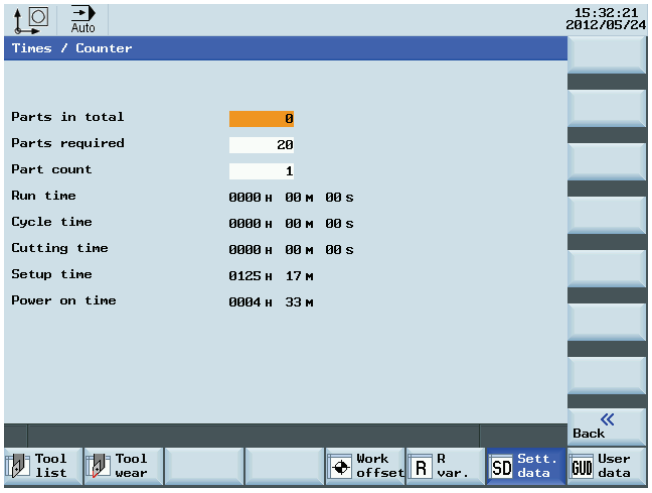
The part counter is available for the SINUMERIK 808D to count the following information:

Time	Corresponding system variable	Description
Required parts	$AC_REQUIRED_PARTS	Required parts to be counted. Activated by setting MD27880 BIT0 = 1: • BIT 1 = 0: if "**Part count**" = "**Parts required**", alarm or interface DB3300.DBX4001.1 = 1
Parts in total	$AC_TOTAL_PARTS	Total number of counted parts. Activated by setting MD27880 BIT 4 = 1: • BIT 5 = 0: M02/M30 increases "**Parts in total**" to "**1**" • BIT 5 = 1: the M code defined by MD27882 increases "**Parts in total**" to "**1**" • BIT 6 = 0/1: the counter does not work when "**Program test**" in active
Part count	$AC_ACTUAL_PARTS	Parts actually counted. Activated by setting MD27880 BIT 8 = 1: • BIT 9 = 0: M02/M30 increases "**Parts in total**" to "**1**" • BIT 9 = 1: the M code defined by MD27882 increases "**Parts in total**" to "**1**" • BIT 10 = 0/1: the counter does not work when "**Program test**" in active

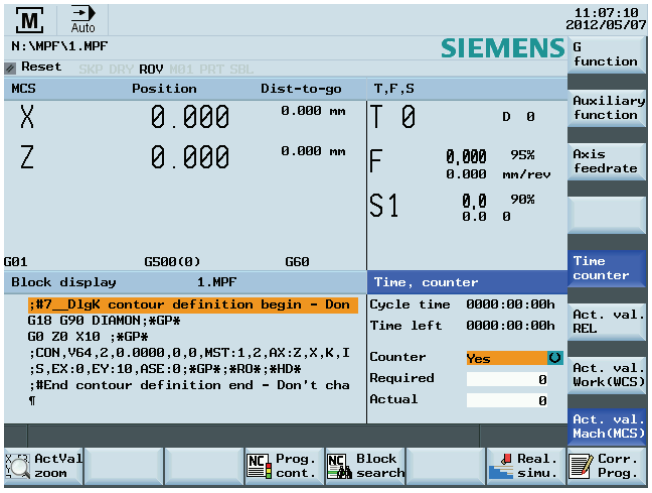
Table 16- 1 Relevant parameters

No.	Name	Value	Descriptions
27880	PART_COUNTER	Actual value	Configuring and activating the part counter
27882	PART_COUNTER_MCODE	Actual value	Defining an M code for the counting action: 0 to 99

In the "OFFSET" operating area, the part counter (horizontal softkey "Sett. data" > Vertical softkey "Time counter") can be displayed counting following information:



Under the "AUTO" mode in the "MACHINE" operating area, the part counter (vertical softkey "Time counter" > <SELECT> key) can also be displayed counting the following information:



Note

All of the numbers that have been entered must be confirmed using the <INPUT> key.

16.4 Prog_Event function

With the Prg_Event function, an asynchronous subroutine program called "PROG_EVENT.SPF" is triggered to be executed at certain states such as the end of a program, NC reset, etc.. You must save the PROG_EVENT.SPF file under the cycle directory (N: \CMA).

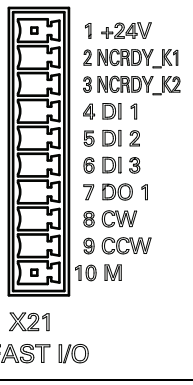
Table 16- 2 Relevant parameters

No.	Name	Value	Descriptions
11450	SEARCH_RUN_MODE	7H	-
20106	PROG_EVENT_IGN_SINGLE BLOCK	1FH	-
20107	PROG_EVENT_IGN_INHIBIT	CH	-
20108	PROG_EVENT_MASK	Actual value	Triggering modes for N: \CMA\PROG_EVENT.SPF: • Bit 0: activating the PROG_EVENT.SPF during the NC commissioning • Bit 1: activating the PROG_EVENT.SPF at the end of a NC program • Bit 2: activating the PROG_EVENT.SPF using the RESET key • Bit 3: activating the PROG_EVENT.SPF after powering up the NC
20109	PROG_EVENT_MASK_PROP ERTIES	1H	-

16.5 Fast I/O

Hardware description

The FAST I/O interface (X21) provides 3 digital inputs and 1 digital output:

Illustration	Pin	Signal	Description	Variable
	4	DI1	Fast input 1 with address DB2900.DBX0.0	$A_IN[1]
	5	DI2	Fast input 2 with address DB2900.DBX0.1	$A_IN[2]
	6	DI3	Fast input 3 with address DB2900.DBX0.2	$A_IN[3]
	7	DO1	Fast output 1 with address DB2900.DBX4.0	$A_OUT[1]

Relevant parameters

MD No.	Name	Meaning	Value
10366	HW_ASSIGN_DIG_FASTIN[0]	Hardware assignment for the fast inputs	10101
10368	HW_ASSIGN_DIG_FASTOUT[0]	Hardware assignment for the fast outputs	10101

PLC interface addresses

DB2900	Signals from fast inputs and outputs							
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0						Input 3	Input 2	Input 1
4								Output 1

Applications of the fast inputs/outputs

Fast inputs

In the PLC application program, you can directly read each bit value from the address **DB2900.DBX0.0**.

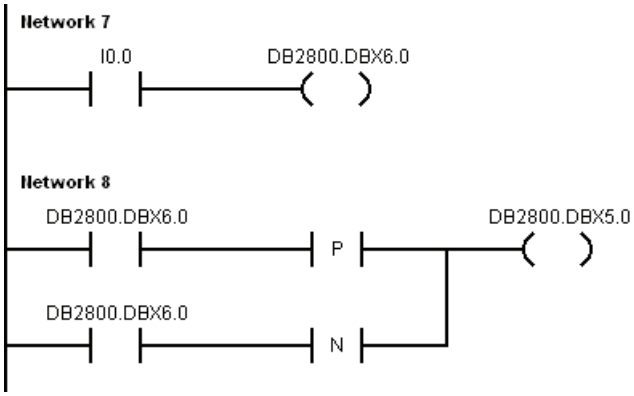
In a part program, you can read each bit value from the address DB2900.DBX0.0 via corresponding system variable.

Fast outputs

From the address **DB2900.DBX4.0** you can not assign a value to the fast output; otherwise, the PLC application program will stop with an error. However, you can assign a value to the fast output from address **DB2800.DBX5.0** and **DB2800.DBX6.0**.

In the PLC application program, you can trigger the address **DB2800.DBX5.0** with a rising edge or a negative edge at the address **DB2800.DBX6.0**, and thus the address **DB2900.DBX4.0** will vary with the address **DB2800.DBX6.0**.

For example, if you want to use **I0.0** to trigger or deactivate the set/reset of the address **DB2900.DBX4.0**, you can write as follows in the PLC application program:



In a part program, you can set or reset the fast output via its corresponding variable. The system variable is **$A_OUT[1]**.

For example:

```
$A_OUT[1]=1 > set DB2900.DBX4.0=1
M30
```

16.6 Creating user cycles

The SINUMERIK 808D is integrated with standard Siemens cycles. If necessary, you can also create your own cycles.

To create a customized cycle, you must prepare the files shown below:

- User cycle file
- Extended user text file
- User cycle alarm file
- User cycle softkey index file
- User cycle parameter file
- User cycle bitmap file

16.6.1 Creating the extended user text file

The extended user text file is required for the display of respective screen texts, cycle messages and softkey texts.

Naming rule

almc_xxx.txt

Here "xxx" refers to the language denotation, for example, eng.

Text definition rules

When defining the texts, you must follow the rule below:

<Identifier> "<Text>" // <# chars & lines>

- <Identifier>: here you define the identifier with a number. The number ranges from 83000 to 84999.
- <Text>: here you define the actual text.
- <# chars & lines>: here you specify the available space for the text in the GUI in number of characters and lines. You can start a new line by inserting the character of "**%n**". A maximum of 2 lines with 9 characters each are available for softkey texts.

Examples

83000 "User%nCycles" // 2*9 ⇒ two lines. Each line with nine characters space

83001 "CYCLE10" // 9 ⇒ one line with nine characters space

16.6.2 Creating the user cycle softkey index file

The user cycle softkey index file (cov.com) file is required to define the softkeys for the user cycle. You can create the cov.com file with a text editor like the WordPad or Notepad.

Text definition rules

Sx.y.z\$+identifier\bitmap(cycle)

Table 16- 3 Definitions of the parameter

Parameters	Value range	Significance
X	5	The fifth horizontal key.
Y	1 to 8	The first to eighth vertical key in the first level.
Z	1 to 8	The first to eighth vertical key in the second level.
\$+identifier\	-	Defined in the cycle text file.
bitmap(cycle)	-	The bitmap for the user cycle. The bitmap name must be followed with name of the user cycle.

Examples

```
S5.0.0\$83000\                  > define a softkey (identifier: 83000) at the horizontal key 5.
S5.1.0\$83001\CN1(CYCLE100)     > define a softkey (identifier: 83001) at the first vertical key of
M17                             the first level when pressing the horizontal key 5.
```

16.6.3 Creating the user cycle parameter file

The user cycle parameter file (sc.com) file is required to define the help information and the parameters for the user cycle. You can create the sc.com file with a text editor like the WordPad or Notepad.

Text definition rules

The "//" symbol indicates the beginning of a cycle description.

If you have created an image to display on the left of the screen at cycle start, call the image at the first line. The image is followed by the cycle name written in brackets.

Now define the parameters for the individual variables according to the format shown in the table below:

Line	Description of the parameters	Entry
1	Start of variable declaration	(
2	Variable type	R - REAL I - INTEGER C - CHAR S - STRING

Line	Description of the parameters	Entry
3	Separator	/
4	• Minimum value + space + maximum value • * + different values for selection	• Minimum value + space + maximum value • * + different characters (use space to separate the different characters) Note that you can also define different pictures for the characters.
5	Separator	/
6	Default value	Value passed in the cycle if no entry is made.
7	Separator	/
8	Help information	$ + the identifier defined in the cycle text file
9	End of variable declaration	)
10	Start of description	[
11	Short text	The text displayed in the parameter screen form (defined in the cycle text file).
12	Separator	/
13	Text in the screen	Text preceding the input screen. A maximum of 5 characters in length.
14	End of description	]
15	Line-specific image	/B name.bmp

Note

Separators, start and end identifiers must always be entered.

The lines 4, 6 and 15 can be left blank.

If no texts are stored with the $identifier, three question marks appear in the associated fields on the screen.

Example

```
//CN1(CYCLE100)
(R/0 99999.999//$83002)[$83003/DIA]
(R/0 99999.999//$83004)[$83005/DIAF]
(R/-9999.999 99999.999//$83004)[$83004/STAP]
(R/-9999.999 99999.999//$83025)[$83005/ENDP]
(R/0 99999.999//$83026)[$83006/MID]
(R/0 99999.999//$83027)[$83007/UX]
(I/*0 1 2/0/$83028)[$83008/MACH]/B CN1
(R/1 99999.999/1/$83029)[$83009/VRT]
```

16.6.4 Creating the user cycle file

You can create a user cycle file according to different machining functions. It is a subroutine program that can be used at calling a cycle.

Naming rule

CYCLExxx.SPF

Here "xxx" refers to the cycle number. It **must not** exceed four digits.

Note

The name of a user cycle **must not** be same with that of a standard Siemens cycle. It is recommend to use a cycle number with the range of 100 to 800.

Programming example

Create the program with a wordpad or notepad.

As a cycle screen always also transfers values as call parameters to the user cycle, the transfer interface is defined as follows.

```
PROC CYCLE100(REAL DIA,REAL DIAF,REAL STAP,REAL ENDP,REAL MID,REAL UX,INT MACH,REAL
VRT) SAVE SBLOF DISPLOF
```

PROC is a keyword followed by the cycle name with the cycle number. All the transfer parameters for the screen are contained within brackets with the data type and name separated by commas.

```
PROC CYCLE100(REAL DIA,REAL DIAF,REAL STAP,REAL ENDP,REAL MID,REAL
UX,INT MACH,REAL VRT) SAVE SBLOF DISPLOF
DEF REAL VAR1
IF $P_EP[X]<DIA GOTOF LL1
LL3:
IF DIAF>DIA GOTOF END2
START:
IF MACH==0 GOTOF ROUGHING1
IF MACH==1 GOTOF FINISHING
IF MACH==2 GOTOF ROUGHING1
DEF REAL VAR1
ROUGHING1:
R101=(DIA-DIAF)/2-UX
R102=R101/MID
R103=TRUNC(R102)
R104=0
VAR1=DIA
IF R103<=1 GOTOF ROUGHING2
LL2:
SBLON
G90 G0 X=VAR1 Z=STAP+2
G1 Z=ENDP
G91 X=MID
G0 G91 X=VRT Z=VRT
G90 G0 Z=STAP+2
SBLOF
VAR1=VAR1-2*MID
R104=R104+1
IF R104<=R103 GOTOB LL2
IF R104>R103 GOTOF ROUGHING2
ROUGHING2:
SBLON
G90 G0 X=DIAF+UX
G1 Z=ENDP
G0 G91X=VRT Z=VRT
G90 G0 X=DIA+2
Z=STAP+2
IF MACH==2 GOTOF FINISHING
SBLOF
RET
FINISHING:
SBLON
G0 X=DIAF
G1 Z=ENDP
G1 X=DIA+VRT
G0 G91X=VRT Z=VRT
G90 Z=STAP+2
SBLOF
RET
LL1:
IF $P_EP[Z]<STAP GOTOF END1
GOTOB LL3
END1:
SETAL(65000)
STOPRE
M0
RET
END2:
SETAL(65001)
STOPRE
M0
RET
```

16.6.5 Creating the user cycle alarm file

The user cycle alarm file is required to display alarm numbers and alarm messages for user cycles.

Naming rule

alc_xxx.txt

Here "xxx" refers to the language denotation, for example, eng.

Text definition rules

When defining the texts, you must follow the rule below:

<AlarmNumber> "<Text>" // <# chars & lines>

- <AlarmNumber>: here you define the alarm number. The number ranges from 65000 to 79999.
- <Text>: here you define the actual alarm text.
- <# chars & lines>: here you specify the available space for the text in the GUI in number of characters and lines. You can start a new line by inserting the character of "**%n**".

Examples

65000 "Current tool position is incorrect" // 34 ⇒ one lines with thirty-four characters space

65001 "DIAF is bigger than DIA" // 23 ⇒ one line with twenty-three characters space

16.6.6 Creating the user cycle bitmap file

The cycle icons **must** be stored as bitmap files (*.bmp) with a maximum size of **224 * 224** pixels in **16** colors.

The icon name **must** begin with an uppercase/lowercase "**C**" and its length **must not** exceed **32** characters including the file extension (e.g. CN1.bmp).

Note

If 16 colors are not sufficient for the display, you can also use 24-bit color depth bitmaps.

16.6.7 Transferring the desired files to the control system

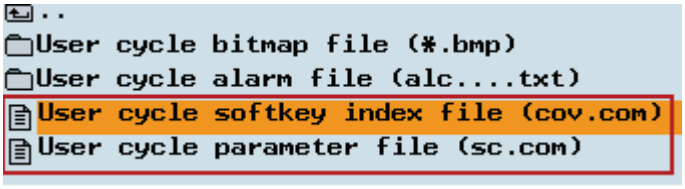
Proceed as follows to transfer the required files to the SINUMERIK 808D control system.

Importing the cov.com file and sc.com file

1. Save the required files on a USB flash disk.
2. Insert the USB flash disk into the USB interface at the front of the PPU.
3. Switch to the "SYSTEM" operating area by pressing 

+ 

.
4. Press the "USB" softkey and multi-select the cov.com file and sc.com file with <SELECT> and copy them with the "Copy" softkey.
5. Press the "808D data" softkey and access the folder "HMI data" > "User cycle files".



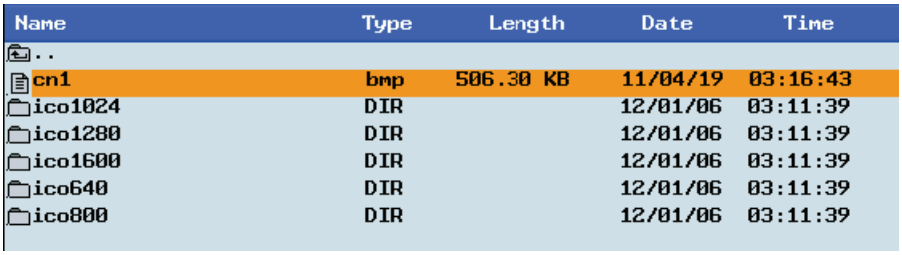
Replace the empty files with the "**Paste**" softkey.

Importing the user cycle alarm file

1. Press the "USB" softkey and select a user cycle alarm file (for example, alc_eng.txt) with the <SELECT> key. Press "Copy".
2. Press the "808D data" softkey. Press <INPUT> to access the "HMI data" > "User cycle files" > "User cycle alarm file (alc....txt)" folder and then press "Paste".

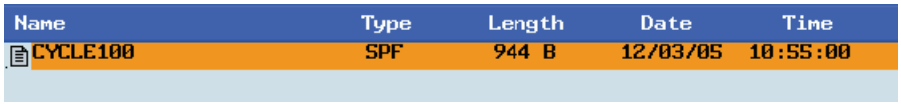
Importing the bitmap file

1. Press the "USB" softkey and select a bitmap file (for example, cn1.bmp) with the <SELECT> key. Press "Copy".
2. Press the "808D data" softkey. Press <INPUT> to access the "HMI data" > "User cycle files" > "User cycle bitmap file (*.bmp)" folder and then press "Paste".



Importing the user cycle file

1. Press the "USB" softkey and select a user cycle file (for example, CYCLE100). Press the "Copy" softkey.
2. Press the "User cycle" softkey and then press "Paste":



Importing the extended user text file

1. Press softkeys "Sys. data" > "USB". Select an extended user text file (for example, almc_eng.txt) and copy it with the "Copy" softkey.
2. Press the "808D data" softkey. Press <INPUT> to access the "HMI data" > "Extended user text file (almc....txt)" folder and then press "Paste".

Note

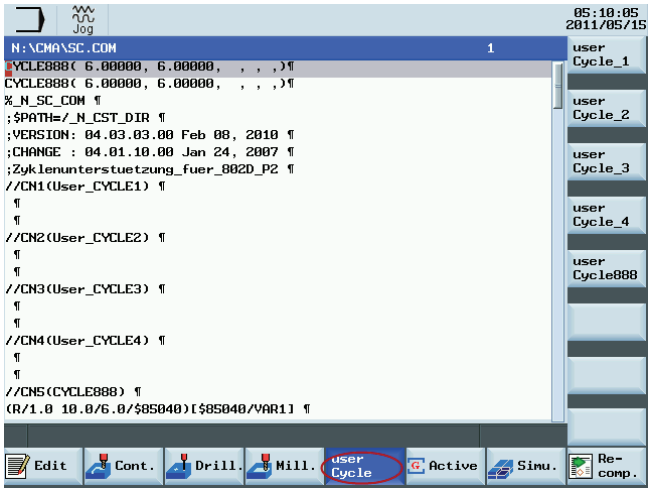
After you press "Paste" to import the cov.com file, sc.com file, alc_xxx.txt file and almc_xxx.txt file, a system message will appear to prompt you to restart the HMI by pressing "OK". Then press "OK" to restart the HMI so that the new data will be available.

16.6.8 Call the created user cycle

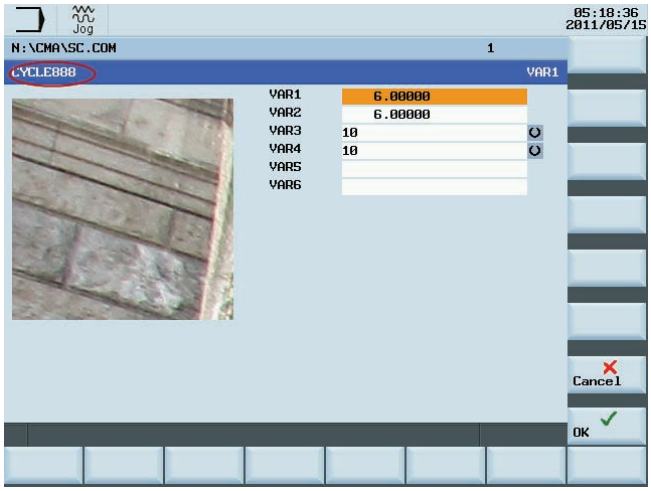
After you transfer all the files necessary for your own cycle to the control system, the cycle is created successfully. Then you can use the cycle in the "PROGRAM" operating area.

Proceed as follows to call the created cycle, for example, CYCLE888.

1. Press <PROGRAM> on the PPU, and now the horizontal softkey "user Cycle" appears. Press "user Cycle".



2. Press the vertical softkey "user Cycle888" to open the "CYCLE888" window.



3. Set the parameters according to your own requirements, and then press "OK" to save the settings. Or, press "Cancel" to quit the cycle.

16.6.9 Editing the user cycle screens

You can edit the softkeys, identifiers, bitmaps or parameters for user cycles.

To do so, export the relevant files and edit them on a PC. After that, import them back to the respective folders and restart the control system.

16.7 Generating user dialogs using customized EasyXLanguage scripts

16.7.1 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 SINUMERIK 808D 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.

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 corresponding 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 Section "XML identifier (Page 232)".

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.

16.7.2 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 (EasyXLanguage scripts) are included in the "...\examples\easyXL" folder of the Toolbox.

An XML editor or another form of text editor can be used to generate the configuration files.

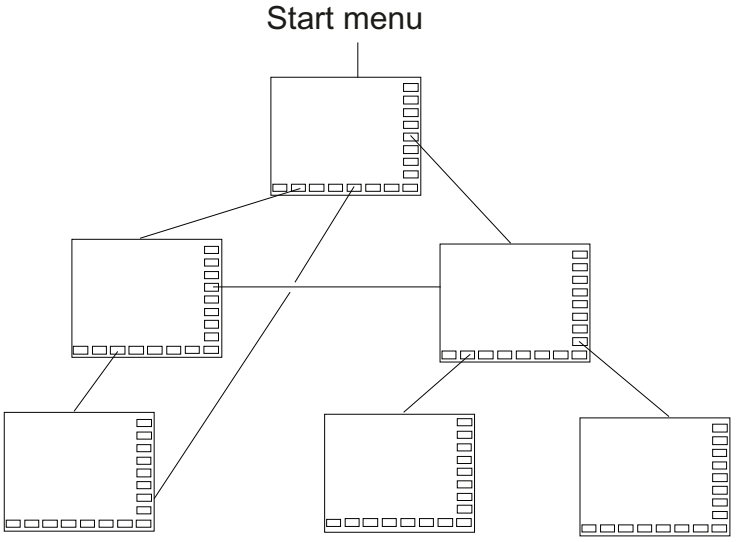
Note

No distinction is made between upper and lower case letters.

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:



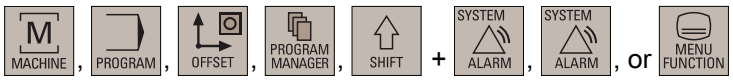
Start menu

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.

Loading your own dialogs or additional softkey bars can be linked with the main menu. Additional actions can be performed using these softkey bars.

Returning to the standard application

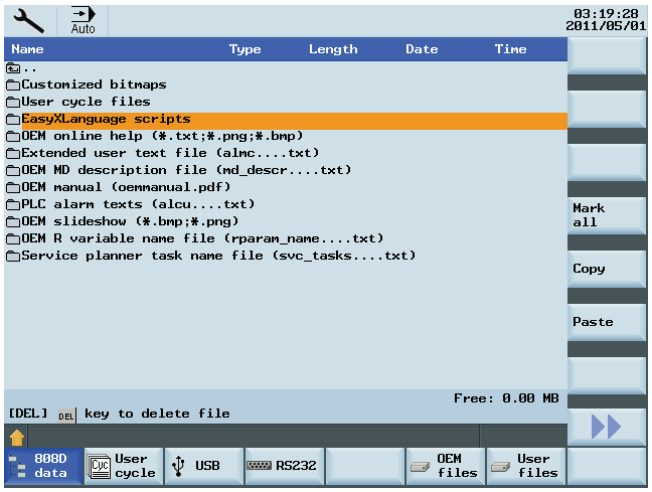
You can exit the newly created user interfaces and return to the standard application by pressing one of the following keys or key combination on the PPU:



16.7.3 Configuration files (EasyXLanguage)

Loading the configuration files

The generated configuration files must be copied from a USB stick to the "SYSTEM" operating area > "Sys. data" > "808D data" > "HMI data" menu > "EasyXLanguage scripts" folder. See the screen below:



Files for configuration

The following files in the control system's "EasyXLanguage scripts" folder are needed to configure the user dialogs:

File type	Name of the file	Meaning
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 "CUSTOM" operating area on the HMI.

File type	Name of the file	Meaning
Text file	"almc.txt"	This text file contains the texts for the menus and dialog forms for individual languages.
Bitmaps	"*.bmp" (E.g., "text.bmp") "*.png" (E.g., "text.png")	Archive with the bitmaps. The control system supports BMP and PNG formats.
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.

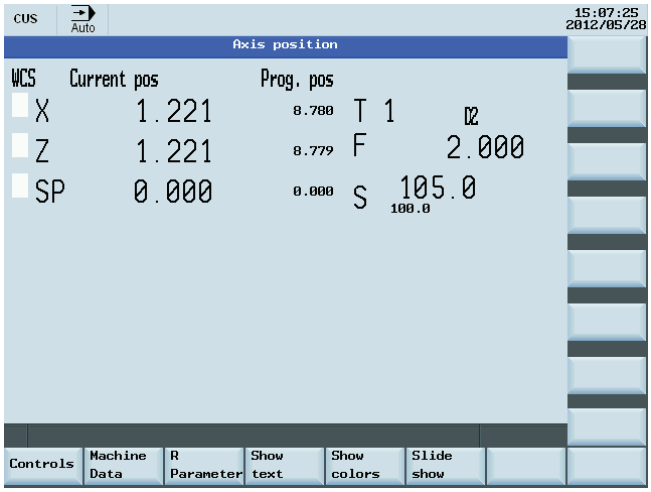
Note

As soon as there is an "xmldial.xml" script file in in the "EasyXLanguage scripts" folder, you can start this user dialog in the "CUSTOM" operating area.

After the initial copying process, the control system must be reset via "SYSTEM" operating area> "Start-up" > "NC" > "Standard 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.



Note

If configured and programmed dialogs need to be used at the same time, the script language must be used to call the programmed dialogs. The functions required for this purpose are described in Section "Predefined functions (Page 265)".

16.7.4 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

16.7.5 Language dependency

Language-dependent texts are used for:

- Softkey labels
- Headers
- Help texts
- Any other texts

The language-dependent texts are stored in the text file (almc.txt).

16.7.6 XML identifier

16.7.6.1 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 following instructions can be used 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 Section "Instruction/identifier description (Page 233)".

16.7.6.2 Instruction/identifier description

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. **Syntax:** `<CONTROL_RESET resetnc="TRUE" />` **Attributes:** • **RESETNC = "TRUE"** The NC component is restarted

Tag identifier	Meaning

DATA	The tag enables the NC, PLC, and GUD data to be directly written to.
	The "Addressing components (Page 246)" section contains details on address formation.
	Attribute:
	• **name**
	Variable address
	Tag value:
	All alphanumeric terms are approved as tag values. If a value is to be written from a local variable directly, the **$** replacement operator preceding the name of the local variable should be used.
	Syntax:
	`<DATA name="<variable name>"> value </DATA>`
	Example:
	`<DATA name = "plc/mb170"> 1 </DATA>`
	…
	`<LET name = "tempVar"> 7 </LET>`
	`<!– the contents of the local variables "tempVar" are written to bit memory byte 170 →`
	`<DATA name = "plc/mb170">$tempVar</DATA>`

DATA_LIST	The tag enables the listed machine data to be saved or restored.
	Up to 20 temporary data lists can be created.
	Attributes:
	• **action**
	read– the values of the listed variables are stored in a temporary memory
	append– the values of the listed variables are added to an existing list
	write– the backed up values are copied to the relevant machine data
	• **id**
	The identifier is used to identify the temporary memory
	Syntax:
	`<DATA_LIST action="<read/write/append>"  id="<list name>">`
	`NC/PLC Address compilation`
	`</DATA_LIST>`
	Example:
	`<DATA_LIST action ="read" id="<name>">`
	`nck/channel/parameter/r[2]`
	`nck/channel/parameter/r[3]`
	`nck/channel/parameter/r[4]`
	`$MN_USER_DATA_INT[0]`
	…
	`</ DATA_LIST>`
	`<DATA_LIST action ="write" id="<name>" />`

ELSE	Instruction for situations where the condition has not been met (**IF, THEN, ELSE**)

Tag identifier	Meaning
FORM	The tag contains the description of a user dialog. The relevant tags are described in the section on generating menus and dialog forms. **Syntax:** `<FORM name="<dialog name>" color="#ff0000">` **Attributes:** • **color** Background color of the dialog form (color coding, see Section Color coding (Page 245)) – Default white • **name** Identifier of the form • **xpos** X-position of the top left corner of the dialog box (optional) • **ypos** Y position of the top left corner (optional) • **width** Extension in the X direction (in pixels) (optional) • **height** Extension in the Y direction (in pixels) (optional)
HMI_RESET	The tag initiates an HMI restart. The interpretation is cancelled after this operation.

Tag identifier	Meaning

IF	Conditional statement (**IF, THEN, ELSE**)
	The THEN and ELSE tags are enclosed in the IF tag.
	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.
	Syntax:
	`<IF>`
	`<CONDITION> Condition != 7 </CONDITION>`
	`<THEN>`
	`Instruction for the case: Condition fulfilled`
	`</THEN>`
	`<ELSE>`
	`Instruction for the case: Condition not fulfilled`
	`</ELSE>`
	`</IF>`
	Example:
	`<IF>`
	`<CONDITION> "plc/mb170" != 7 </CONDITION>`
	`<THEN>`
	`<OP> "plc/mb170" = 7 </OP>`
	…
	`</THEN>`
	`<ELSE>`
	`<OP> "plc/mb170" = 2 </OP>`
	…
	`</ELSE>`
	`</IF>`

INCLUDE	The instruction includes an XML description. (see also DYNAMIC_INCLUDE in this table) **Attribute:** • **src** Contains the path name. **Syntax:** `<?INCLUDE src="<Path name>" ?>`

Tag identifier	Meaning
LET	The instruction creates a local variable under the specified name. **Fields:** Using the attribute **dim** (dimension) single or two-dimensional fields can be created. The field index is used to address the individual field elements. For a two-dimensional field, initially the line index is specified and then the column index. • Single-dimensional field: Indices 0 to 4 • Two-dimensional field: Index line 0 to 3 and index column 0 to 5 **Attributes:** • **name** Variable name • **type** 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. `<LET name = "VAR1" type = "INT" />` • **permanent** If the attribute is set to **true**, then the variable value is saved permanently. This attribute is only effective for a global variable. • **dim** 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. A field element is accessed via the field index, which is specified in square brackets after the variable name. name[index] or name[row,column] – Single-dimensional field: dim="<Number of elements>" – Two-dimensional field: dim="<Number of lines>,<number of columns>" Non-initialized field elements are pre-assigned with "0".

Tag identifier	Meaning

LET Continued	**Example:**
	Single-dimensional field:
	`<let name="array" dim="10"></let>`
	Two-dimensional field:
	`<let name="list_string" dim="10,3" type="string"></let>`
	Pre-assignment:
	A variable can be initialized with a value.
	`<LET name = "VAR1" type = "INT"> 10 </LET>`
	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.
	• Pre-assignment for a string variable:
	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> , the characters "\\n" should be added at the end of the line.
	`<LET name = "text" type = "string"> F4000 G94\\n`
	`G1 X20\\n`
	`Z50\\n`
	`M2\\n`
	`</LET>>`
	`Fields (Arrays):`
	`<let name="list" dim="10,3">`
	`{1,2,3},`
	`{1,20}`
	`</let>`
	`<let name="list_string" dim="10,3" type="string">`
	`{"text 10","text 11"},`
	`{"text 20","text 21"}`
	`</let>`
	Assignment:
	Values made up of the machine data or subroutines can be assigned to a variable using the assignment operation "=".
	A variable remains valid until the end of the higher-level XML block.
	Variables which are to be available globally should be created directly after the **DialogGUi** tag.
	The following must be observed for a dialog box:
	• The message processing opens the corresponding tag.
	• The tag is closed after the message has been executed.
	• All variables within the tag are deleted when closing.

MSG	The operator component shows the message which is indicated in the tag. If an alarm number is used, the dialog box displays the text which is saved for the number. **Example:** `<MSG text ="my message" />`

Tag identifier	Meaning

MSGBOX	The instruction opens a message box whose return value can be used for branching.
	Syntax:
	`<MSGBOX text="<Message>" caption="<caption>" retvalue="<variable name>" type="<button type>" />`
	Attributes:
	• **text**
	Text
	• **caption**
	Header
	• **retvalue**
	Name of the variables to which the return value is copied:
	1 – OK
	0 – CANCEL
	• **type**
	Acknowledgement options:
	"BTN_OK"
	"BTN_CANCEL"
	"BTN_OKCANCEL"
	If an alarm number is used for the **"text"** or **"caption"** attribute, the message box displays the text which is saved for the number.
	Example:
	`<MSGBOX text="Test message" caption="Information" retvalue="result" type="BTN_OK" />`

Tag identifier	Meaning

OP	The tag executes the specified operations.
	For the purpose of accessing the NC, PLC, and drive data, the complete variable name should be placed in **quotation marks**.
	`PLC: "PLC/MB170"`
	`NC: "NC/Channel/…"`
	Example:
	`<LET name = "tmpVar" type="INT"> </LET>`
	`<OP> tmpVar = "plc/mb170" </OP>`
	`<OP> tmpVar = tmpVar *2 </OP>`
	`<OP> "plc/mb170" = tmpVar </OP>`
	Character string processing:
	The operation instruction is able to process character strings and assign the results to the string variable specified in the equation.
	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.
	Example:
	`<LET name="buffer" type="string"></LET>`
	`…`
	`…`
	`…`
	`<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>`

PASSWORD	The tag opens a dialog for entering the password.
	Once the entry has been confirmed, the character string is available in the specified reference variable.
	Syntax:
	`<PASSWORD refVar ="<variable name>" />`
	Attribute:
	• **refVar**
	Name of the reference variable
	Example:
	`<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.
	If the text contains formatting identifiers, the variable values are inserted at the appropriate places.
	Syntax:
	`<PRINT name="Variable name " text="text %Formatting "> Variable, … </PRINT>`
	`<PRINT text="text %Formatting"> Variable, … </PRINT>`
	Attributes:
	• **name**
	Name of the variable where the text is to be stored (optional)
	• **text**
	Text
	Formatting:
	The character "%" causes the variable specified as the value to be formatted.
	%[Flags] [Width] [.decimal places] type
	• Flags:
	Optional character for defining output formatting:
	– Right-justified or left-justified ("-") for left-justified
	– Add leading zeros ("0")
	– Fill with blanks
	• Width:
	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.
	• 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 for the print instruction. These characters need to be specified.
	– d: Integer value
	– f: Floating point number
	– s: String

Tag identifier	Meaning

PRINTContinued	**Values:**
	Number of variables whose values are to be inserted into the text.
	The variable types must match the corresponding type identifier for the formatting instruction and must be separated from one another using a comma.
	Example:
	Output of a text in the information line
	`<PRINT text="Infotext" />`
	Output of a text with variable formatting
	`<LET name="trun_dir"></LET>`
	`<PRINT text="M%d">trun_dir</PRINT>`
	Output of a text in a string variable with variable formatting
	`<LET name="trun_dir"></LET>`
	`<LET name="str" type="string" ></LET>`
	`<print name="str" text="M%d ">trun_dir</print>`

STOP	Interpretation is cancelled at this point.

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.
	The **DEFAULT** instruction is executed when none of the constants match the expression.
	Syntax:
	`<SWITCH>`
	`<CONDITION> Value </CONDITION>`
	`<CASE value="Constant 1">`
	`Instructions`
	`…`
	`</CASE>`
	`<CASE value="Constant 2">`
	`Instructions`
	`…`
	`</CASE>`
	`<DEFAULT>`
	`Instructions`
	`…`
	`</DEFAULT>`
	`</SWITCH>`

THEN	Operation, if the condition has been fulfilled (**IF, THEN, ELSE**)

Tag identifier	Meaning

FOR	For loop
	`for (initialization; test; continuation) instruction(s)`
	Syntax:
	`<FOR>`
	`<INIT>…</INIT>`
	`<CONDITION>…</CONDITION>`
	`<INCREMENT>…</INCREMENT>`
	`Instructions`
	…
	`</FOR>`
	The For loop is executed as follows:
	1. Evaluation of the term **initialization** (INIT).
	2. Evaluation of the term **test** (CONDITION) as a Boolean term.
	If the value is false, the For loop is ended.
	3. Execution of the following instructions.
	4. Evaluation of the term **continuation** (INCREMENT)..
	5. Continue with 2.
	All the variables used within the INIT, CONDITION, and INCREMENT branches should be created outside the FOR loop.
	Example:
	`<LET name = "count">0</LET>`
	`<FOR>`
	`<INIT>`
	`<OP> count = 0</OP>`
	`</INIT>`
	`<CONDITION> count <= 7 </CONDITION>`
	`<INCREMENT>`
	`<OP> count = count + 1 </OP>`
	`</INCREMENT>`
	`<OP> "plc/qb10" = 1+ count </OP>`
	`</FOR>`

WAITING	The tag waits for the component to undergo a hot restart after an NC reset. **Attributes:** • **WAITINGFORNC** = "TRUE" - the system waits for the NC to restart **Syntax:** `<WAITING WAITINGFORNC = "TRUE"/>`

Tag identifier	Meaning

WHILE	WHILE loop
	`WHILE (Test)`
	`  Instruction`
	Syntax:
	`<WHILE>`
	`<CONDITION>…</CONDITION>`
	`Instructions`
	`…`
	`</WHILE>`
	The While loop is used to execute a sequence of instructions repeatedly while a condition is met. This condition is tested before the sequence of instructions is executed.
	Example:
	`<WHILE>`
	`<CONDITION> "plc/ib9" == 0 </CONDITION>`
	`<DATA name = "PLC/qb11"> 15 </DATA>`
	`</WHILE>`

DO_WHILE	Do while loop
	`DO`
	`  Instructions`
	`WHILE (Test)`
	Syntax:
	`<DO_WHILE>`
	`Instructions`
	`…`
	`<CONDITION>…</CONDITION>`
	`</DO_WHILE>`
	The Do while loop comprises a block of instructions 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.
	Example:
	`<DO_WHILE>`
	`<DATA name = "PLC/qb11"> 15 </DATA>`
	`<CONDITIION> "plc/ib9" == 0 </CONDITION>`
	`</DO_WHILE>`

16.7.6.3 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"

16.7.6.4 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
<	<
>	>
&	&
"	"
'	'

16.7.6.5 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.

16.7.7 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.

16.7.7.1 PLC addressing

Addressing the PLC starts with the path section **plc**.

Table 16- 4 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 16- 5 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>
```

16.7.7.2 NC variable addressing

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 808D Parameter Manual.

Example:

```
<LET name = "tempStatus"></LET>
<OP> tempStatus ="nck/channel/state/chanstatus" </OP>
```

16.7.7.3 Addressing machine and setting data

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>
```

16.7.7.4 Addressing the 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>
```

16.7.8 Generating user menus

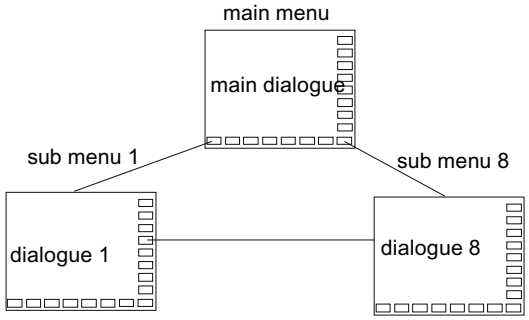
16.7.8.1 Generating softkey menus and dialog forms

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>
```



Tag identifier	Meaning
FORM	This tag contains the description of a user dialog. **Attributes:** • **color** Background color of the dialog box (color coding, see Chapter, color coding) • **name** Identifier of the form • **xpos** X-position of the top left corner of the dialog box (optional) • **ypos** Y position of the top left corner (optional) • **width** Extension in the X direction (in pixels) (optional) • **height** Extension in the Y direction (in pixels) (optional) **Dialog messages:** • **INIT** • **PAINT** • **TIMER** • **CLOSE** • **FOCUS_IN**

FORM continued	**Syntax:**
	`<FORM name = "<dialog name>" color = "#ff0000">`
	Example:
	`<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"`
	` </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>`
	
	` </PAINT>`
	
	`</FORM>`

Tag identifier	Meaning
INIT	**Dialog box message** 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.

FOCUS_IN	**Dialog box message**
	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.
	This message can be used, for example, to output images depending on the focus position.
	Example:
	`<focus_in>`
	`<PRINT text="focus on filed:%s, %d">$focus_name,`
	`$focus_item_data </PRINT>`
	`</focus_in>`

PAINT	**Dialog box message** 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. 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** The tag is executed cyclically. Each form is assigned a timer that initiates that the timer - tag is executed approx. every 100 ms.

CAPTION	The tag contains the title of the dialog box.
	This tag should be used within the INIT tag.
	Syntax:
	`<CAPTION>Titel</CAPTION>`
	Example:
	`<CAPTION>my first dialogue</CAPTION>`

CLOSE	**Dialog box message** This tag is executed before the dialog box is closed.

CLOSE_FORM	The tag closes the active dialog.
	This instruction is only necessary if it involves a cycle dialog that is used in the program editor area. Generally, dialogs are automatically managed and do not have to be explicitly closed.
	Syntax:
	`<CLOSE_FORM/>`
	Example:
	`<softkey_ok>`
	`<caption>OK</caption>`
	`<CLOSE_FORM />`
	`<navigation>main_menu</navigation>`
	`</softkey_ok>`

Tag identifier	Meaning
CONTROL	The tag is used to generate control elements. **Syntax:** `<CONTROL name = "<control name>" xpos = "<X position>" ypos = "<Y position>" refvar = "<NC variable>" hotlink = "true" format = "<format>" />` **Attributes:** • **name** Identifier of the field. The identifier simultaneously represents a local variable, and must not be used a multiple number of times in the form. • **xpos** X position of the top left corner • **ypos** Y position of the top left corner • **fieldtype** Field type If no type is specified, the field is set as an edit field. – **edit** Data can be changed – **readonly** Data cannot be changed **combobox** The field displays the corresponding identifiers instead of numerical values. If the field type "**combobox**" is selected, then the expressions to be displayed must also be assigned to the field. The <ITEM> TAG should be used for this purpose. The combo box saves the index of the currently selected text in the variable belonging to the control (see the attribute **refvar**). – **progressbar** A progress bar with a value range of 0 to 100 appears. 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** – **listbox** The field type generates an empty list box control. Using the tag <ITEM> a list box element can be inserted in the list box. The ITEM attribute **value** allows this element to be assigned a unique value. For example, this can be used to identify the element. Parameters **width** and **height** specify the width and height of the list box. After the control has been created, additional list box elements can be inserted using the functions **AddItem**, **InsertItem** or **LoadItem**. – **graphicbox** The field type generates a 2d broken line graphic control. Using the tag <ITEM> a graphic element can be inserted into the control. Parameters **width** and **height** specify the width and height of the box. **Note:** This control is not linked into the clipping. This means that other elements can cover this control. After the control has been created, additional elements can be inserted using the functions **AddItem** or **InsertItem**. The parameter **itemdata** is not evaluated for this control. – **itemlist** The field type generates a static control, which displays the corresponding identifier instead of numerical values. The <ITEM> tag can be used to assign an identifier to the field. • **item_data** 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. • **refvar** Identifier of the reference variable that can be linked to the field (optional). • **hotlink** = "TRUE" " If the value of the reference variable changes, then the field is automatically updated (optional). • **format** The attribute defines the display format of the specified variable. Formatting data, see print-Tag (optional).

Tag identifier	Meaning
CONTROL continued	**Attributes:** • **time** specifies the data refresh rate (optional). The following specifications are possible: – **super fast** Refresh time < 100 ms – **fast** Refresh time approx. 100 ms – **normal** Refresh time approx. 200 ms – **slow** Refresh time approx. 500 ms • **font** The attribute defines the font size used. – 0: 8*8 – 1: 16*8 – 2: 24*16 (only numbers) – 3: 8*8 double the character height – 4: 16*8 double the character height – 5: 24*16 double the character height (only numbers) • **color_bk** The attribute sets the background color of the control. • **color_fg** The attribute sets the foreground color of the control. ("color coding" see Section "Color coding (Page 245)") • **display_format** 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. The following data formats are permitted: – FLOAT – INT – DOUBLE – STRING Assigning expressions (e.g. text or graphic element to be displayed) to a list box, graphics box or combo box: **Syntax:** <ITEM>Expression</ITEM> <ITEM value ="<Value>">Expression</ITEM>

Tag identifier	Meaning

CONTROL continued	**Example:**
	`<CONTROL name = "button1" xpos = "10" ypos = "10" fieldtype = "combobox ">`
	`<ITEM>text1</ITEM>`
	`<ITEM>text2</ITEM>`
	`<ITEM>text3</ITEM>`
	`<ITEM>text4</ITEM>`
	`</CONTROL>`
	If any integer value is to be assigned to an expression, the attribute **value = "value"** should be added to the tag.
	Rather than consecutive numbers, the control variable now contains the item's assigned value.
	Example:
	`<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>
	Example of a progress bar:
	`<CONTROL name = "progress1" xpos = "10" ypos = "10" width = "100" fieldtype = "progressbar" hotlink = "true" refvar = "nck/Channel/GeometricAxis/actProgPos[1]">`
	`<PROPERTY min = "0" />`
	`<PROPERTY max = "1000" />`
	`</CONTROL>`
	Example, list box:
	`<let name="item_string" type="string"></let>`
	`<let name="item_data" ></let>`
	
	`<CONTROL name="listbox1" xpos = "360" ypos="150" width="200" height="200" fieldtype="listbox" />`
	• Adding elements:
	Elements are added using the function **additem** or **loaditem**.
	• Deleting the content:
	The content is deleted using the function **empty**.
	
	`<op> item_string = _T"text1\\n" </op>`
	`<function name="control.additem">_T"listbox1", item_string, item_data </function>`
	`<op> item_string = _T"text2\\n" </op>`
	`<function name="control.additem">_T"listbox1", item_string, item_data </function>`

Tag identifier	Meaning
CONTROL continued	**Example, graphic box:** `<CONTROL name= "graphic" xpos = "8" ypos="23" width="300" height="352" fieldtype="graphicbox" />` • Adding elements: Elements are added using the function **additem** or **loaditem**. The following 2d elements can be used: – Line - l(inc) – Circle sector - c(ircle) – Point - p(oint) **Structure of an element:** <Element type>; coordinates • Line: l; xs; ys; xe, ye l - line marking Xs - X start position Ys - Y start position Xe - X end position Ye - Y end position • Circle: C, xs, ys, xe, ye, cc_x, cc_y, r C - circular sector marking Xs - X start position Ys - Y start position Xe - X end position Ye - Y end position Cc_x – X coordinate, circle center point CC_y – Y coordinate circle center point • Radius: R • Point: P, x, y P - point marking X - X position Y - Y position • Deleting the graphic: The content is deleted using the function **empty**.

Tag identifier	Meaning

CONTROL continued	**Example:**
	`<let name="item_string" type="string"></let>`
	`<let name="s_z" type="double">100</let>`
	`<let name="s_x" type="double">50</let>`
	`<let name="itemdata"></let>`
	…
	…
	`<control name= "gbox" xpos = "6" ypos="24"  width="328" height="356" fieldtype="graphicbox" />`
	…
	…
	`<print name ="item_string" text="p; %f; %f">s_z, s_x</print>`
	`<function name="control.additem">_T"gbox", item_string, itemdata</function>`
	…
	
	Example itemlist:
	`<CONTROL name = "itemlist1" xpos = "10" ypos = "10" fieldtype = " itemlist">`
	`<ITEM value = "10">text1</ITEM>`
	`<ITEM value = "20">text2</ITEM>`
	`<ITEM value = "12">text3</ITEM>`
	`<ITEM value = "1">text4</ITEM>`
	`</CONTROL>`

HELP_CONTEXT	This tag defines the help topic to be called. It should be programmed in the INIT block.
	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.
	Sequence when activating the help system:
	1. Press the "Info" key.
	2. The dialog supplies the expression "my_dlg_help".
	3. Parser converts the expression into "XmlUserDlg_my_dlg_help" .
	4. Activating the help system.
	5. Submitting the search term "XmlUserDlg_my_dlg_help".
	Syntax:
	`<HELP_CONTEXT name="<context name>" />`
	Example:
	...
	`<INIT>`
	...
	`<CAPTION>my dialogue</CAPTION>`
	`<HELP_CONTEXT name="my_dlg_help" />`
	...
	`</INIT>`

Tag identifier	Meaning
DATA_ACCESS	The tag controls the behavior of the dialog forms when user inputs are being saved. The behavior should be defined within the INIT tag. If the tag is not used, inputs are buffered in each case. Exception: The attribute **hotlink** is set to **true** . **Attribute::** • **type** = "TRUE" – the input values are not buffered. The dialog form copies the input values to the reference variables directly. • **type =** "FALSE" – the values are only copied to the reference variable with the **UPDATA_DATA** type = "FALSE" tag. **Example:** `<DATA_ACCESS type = "true" />`

MENU	The tag defines a menu containing the softkey description and the dialog to be opened.
	Attribute::
	• **name**
	Menu name
	Syntax:
	`<MENU name = "<menu name>">`
	…
	`<open_form …>`
	…
	`<SOFTKEY …>`
	`</SOFTKEY>`
	`</MENU>`

NAVIGATION	This tag defines the menu to be called. This tag can only be set within a softkey block.
	Syntax:
	`<NAVIGATION>menu name</NAVIGATION>`
	Example:
	`<menu name = "main">`
	`<softkey POSITION="1">`
	`<caption>sec. form</caption>`
	`<navigation>sec_menu</navigation>`
	`</softkey>`
	`</menu>`
	
	`<menu name = "sec_menu">`
	`<open_form name = "sec_form" />`
	`<softkey_back>`
	`<navigation>main</navigation>`
	`</softkey_back>`
	</menu>

Tag identifier	Meaning

OPEN_FORM	The tag opens the dialog form given under the name.
	Attribute::
	• **name**
	Name of the dialog form
	Syntax:
	`<OPEN_FORM name = "<form name>" />`
	Example:
	`<menu name = "main">`
	`<open_form name = "main_form" />`
	`<softkey POSITION="1">`
	`<caption>main form</caption>`
	`<navigation>main</navigation>`
	`</softkey>`
	`</menu>`
	
	`<form name="main_form">`
	`<init>`
	`</init>`
	
	`<paint>`
	`</paint>`
	
	`</form>`

PROPERTY	This tag can be used to define additional properties for an operator control.
	Attributes:
	• **max**= "<maximum value>"
	• **min**= "<minimum value>"
	• **default** = "<pre-assignment>"
	• **factor** = "conversion factor"
	• **color_bk**= "<background color coding>"
	• **color_fg**= "<font color coding>"
	• **font** = "<Font number>"
	• **password** = "<true>" - entered character is displayed with "*"
	• **multiline** = "<true>" - permits multi-line inputs in an edit control
	• **disable** = "<true/false>" - locks/permits the input in an edit control
	Example:
	`<CONTROL name = "progress1" xpos = "10" ypos = "10" width = "100"`
	`fieldtype = "progressbar" hotlink = "true" refvar =`
	`"nck/Channel/GeometricAxis/actProgPos[1]">`
	`<PROPERTY min = "0" />`
	`<PROPERTY max = "1000" />`
	`</CONTROL>`
	`<CONTROL name = "edit1" xpos = "10" ypos = "10">`
	`<PROPERTY min = "20" />`
	`<PROPERTY max = "40" />`
	`<PROPERTY default = "25" />`
	`</CONTROL>`

Tag identifier	Meaning

SOFTKEY	The tag defines the properties and responses of a softkey.
	Attributes:
	• **position**
	Number of the softkey. 1-8 horizontal softkeys, 9-16 vertical softkeys
	
	The following additional actions can be defined within the softkey block:
	• **caption**
	• **navigation**
	• **update_controls**
	• **function**
	Syntax:
	`<softkey position = "<1>">`
	…
	…
	`</softkey>`

TEXT	The tag is used to display a text in the specified position. If an alarm number is used, the dialog box displays the text which is saved for the number. **Syntax:** `<TEXT xpos = "<X position>" ypos = "<Y position>"> Text </TEXT>` **Attributes:** • **xpos** X position of the top left corner • **ypos** Y position of the top left corner • **color** Text color (color coding) **Value:** 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.
	Syntax:
	`<IMG xpos = "<X position>" ypos = "<Y position>" name = "<name>" />`
	Attributes:
	• **xpos**
	X position of the top left corner
	• **ypos**
	Y position of the top left corner
	• **name**
	complete path name
	• **transparent**
	Transparent color of the bitmap (see Chapter "Color coding")
	Optional:
	If the image display is to differ from the original size, the dimensions can be defined using the attributes **width** and **height**.
	• **width**
	Width in pixels
	• **height**
	Height in pixels
	Examples:
	`<IMG xpos = "20" ypos = "40" name = "f:/appl/test.bmp" />`
	`<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. **Syntax:** `<BOX xpos = "<X position>" ypos = "<Y position>" width = "<X extension>" height = "<Y extension>" color = "<Color code>" />` **Attributes:** • **xpos** X position of the top left corner • **ypos** Y position of the top left corner • **width** Extension in X direction (in pixels) • **height** Extension in Y direction (in pixels) • **color** Color coding (for details on color coding, see chapter, Color coding)

Tag identifier	Meaning

FUNCTION	**Function call**
	The tag executes the function body, which is specified under the attribute "**name**.
	Attributes:
	• **name**= "Name of the function body"
	• **return** = "Variable name for saving the result of the function"
	Values:
	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.
	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.
	Syntax:
	`<FUNCTION name = "<function name>" />`
	Calling function expects a return value
	`<FUNCTION name = "<function name>" return = "<Variablenname>" />`
	Parameter transfer
	`<FUNCTION name = "<function name>"> var1, var2, var3 </FUNCTION>`
	`<FUNCTION name = "<function name>"> _T"Text", 1.0, 1 </FUNCTION>`
	Examples:
	See "FUNCTION_BODY".

Tag identifier	Meaning

FUNCTION_BODY	**Function body**
	The tag contains the function body of a subfunction. The function body needs to be programmed within the DialogGui tag.
	Attributes:
	• **name** = "Name of the function body"
	• **parameter** = "Parameter list" (optional)
	The attribute lists the transfer parameters that are required. The parameters must be separated by a comma.
	When the function body is called, the values of the parameters specified in the function call are copied to the transfer parameters listed.
	• **return** = "true"
	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.
	Syntax:
	Function body without parameter
	`<FUNCTION_BODY name = "<function name>">`
	…
	…
	…
	`</ FUNCTION_BODY>`
	Function body with parameter
	`<FUNCTION_BODY name = "<function_name>" parameter = "<p1, p2, p3>">`
	…
	`<LET name = "tmp></LET>`
	`<OP> tmp = p1 </OP>`
	…
	`</FUNCTION_BODY>`
	Function body with return value
	`<FUNCTION_BODY name = "<function_name>" parameter = "<p1, p2, p3>" return = "true">`
	…
	`<LET name = "tmp></LET>`
	`<OP> tmp = p1 </OP>`
	…
	`<OP> $return = tmp </OP>`
	`</FUNCTION_BODY>`

Tag identifier	Meaning

FUNCTION_BODY
continued
	Example:
	`<function_body name = "test" parameter = "c1,c2,c3" return = "true">`
	`<LET name = "tmp">0</LET>`
	`<OP> tmp = c1+c2+c3 </OP>`
	`<OP> $return = tmp </OP>`
	`</function_body>`
	…
	…
	…
	`<LET name = "my_var"> 4 </LET>`
	`<function name = "test" return = " my_var "> 2, 3,4</function>`
	`<print text = "result = %d"> my_var </print>`
	…
	…

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. 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 This tag is only processed within the INIT operation. **Attribut:e:** • **name** Address identifier **Syntax:** `<REQUEST name = "<NC-Variable>" />`

UPDATE_CONTROLS	The tag runs a comparison between the operator controls and the reference variables.
	Attribute::
	• **type**
	The attribute defines the direction of the data comparison.
	= TRUE – data is read from the reference variables and copied to the operator controls.
	= FALSE – Data is copied from the operator controls to the reference variables.
	Syntax:
	`<UPDATE_CONTROLS type = "<Direction>"/>`
	Example:
	`<SOFTKEY_OK>`
	`< UPDATE_CONTROLS type="false"/>`
	`</SOFTKEY_OK>`

16.7.8.2 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>
```

16.7.9 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	Meaning

String.cmp	Two strings are compared with one another from a lexicographical perspective.
	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.
	Parameter:
	str1 - string
	str2 - comparison string
	Syntax:
	`<function name="string.cmp" return ="<int var>" > str1, str2 </function>`
	Example:
	`<let name="rval">0</let>`
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string">A brown bear hunts a brown dog.</let>`
	
	`<function name="string.cmp" return="rval"> str1, str2 </function>`
	Result:
	rval= 0

Function name	Meaning

String.icmp	Two strings are compared from a lexicographical perspective (the comparison is not case-sensitive).
	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.
	Parameter:
	str1 - string
	str2 - Comparison string
	Syntax:
	`<function name="string.icmp" return ="<int var>" > str1, str2 </function>`
	Example:
	`<let name="rval">0</let>`
	
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string">A brown Bear hunts a brown Dog.</let>`
	`<function name="string. icmp" return="rval"> str1, str2 </function>`
	Result:
	rval= 0

String left	The function extracts the first nCount character from string 1 and copies this to the return variable.
	Parameter:
	str1 - String
	nCount - Number of characters
	Syntax:
	`<function name="string.left" return="<result string>"> str1,  nCount </function>`
	Example:
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string"></let>`
	
	`<function name="string. left" return="str2"> str1, 12 </function>`
	Result:
	str2="A brown bear"

Function name	Meaning

String.right	The function extracts the last nCount character from string 1 and copies this to the return variable.
	Parameter:
	str1 - String
	nCount - Number of characters
	Syntax:
	`<function name="string.right" return="<result string>"> str1, nCount </function>`
	
	Example:
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string"></let`
	
	`<function name="string. right " return="str2"> str1, 10 </function>`
	Result:
	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.
	Parameter:
	str1 - string
	iFirst - start index
	nCount - number of characters
	Syntax:
	`<function name="string.middle" return="<result string>"> str1, iFirst, nCount </function>`
	Example:
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string"></let`
	
	`<function name="string. middle " return="str2"> str1, 2, 5 </function>`
	Result:
	str2="brown"

Function name	Meaning

String.length	The function gives the number of characters in a string.
	Parameter:
	str1 - string
	Syntax:
	`<function name="string.length" return="<int var>"> str1 </function>`
	Example:
	`<let name="length">0</let>`
	
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<function name="string.length" return="length"> str1 </function>`
	Result:
	length = 31

Strings.replace	The function replaces all the substrings found with the new string.
	Parameter:
	string - string variable
	find string - string to be replaced
	new string - new string
	Syntax:
	`<function name="<string.replace>"> string, find string, new string </function>`
	Example:
	`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`
	
	`<function name="string.replace" > str1, _T"a brown dog" , _T"a big salmon"</function>`
	Result:
	str1 = "A brown bear hunts a big salmon!"

String.remove	The function removes all the substrings found.
	Parameter:
	string - string variable
	remove string - substring to be deleted
	Syntax:
	`<function name="string.remove"> string, remove string </function>`
	Example:
	`<let name="index">0</let>`
	
	`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`
	`<function name="string.remove" > str1, _T"a brown dog" </function`
	Result:
	str1 = "A brown bear hunts"

Function name	Meaning

Strings.insert	The function inserts a string at the index specified.
	Parameter:
	string - string variable
	index - index (zero based)
	insert string - string to be inserted
	Syntax:
	`<function name="string.insert"> string, index, insert string </function>`
	Example:
	`<let name="str1" type="string">A brown bear hunts. </let>`
	`<let name="str2" type="string">a brown dog</let>`
	
	`<function name="string.insert" > str1, 19, str2</function>`
	Result:
	str1 = "A brown bear hunts a brown dog"

String delete	The function deletes the defined number of characters starting from the start position specified.
	Parameter:
	string - string variable
	start index - start index (zero based)
	nCount - number of characters to be deleted
	
	Syntax:
	`<function name="string.delete"> string, start index , nCount </function>`
	
	Example:
	`<let name="str1" type="string">A brown bear hunts. </let>`
	
	`<function name="string.delete" > str1, 2, 5</function>`
	Result:
	str1 = "A bear hunts"

Function name	Meaning

String.find	The function searches the transferred string for the first match with the substring.
	If the substring is found, the function provides the index to the first character (starting with zero) or, failing this, -1.
	Parameter:
	string - string variable
	findstring - string to be found
	Syntax:
	`<function name="string.find" return="<int val>"> str1, find string </function>`
	Example:
	`<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>`
	Result:
	Index = 2

String.reversefind	The function searches the transferred string for the last match with the substring.
	If the substring is found, the function provides the index to the first character (starting with zero) or, failing this, -1.
	Parameter:
	string - string variable
	find string - string to be found
	Syntax:
	`<function name="string.reversefind" return="<int val>"> str1, find string </function>`
	Example:
	`<let name="index">0</let>`
	`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`
	`<function name="string.reversefind" return="index"> str1, _T"brown" </function>`
	Result:
	Index = 21

String.trimleft	The function trims the starting characters from a string.
	Parameter:
	str1 - string variable
	Syntax:
	`<function name="string.trimleft" > str1 </function>`
	Example:
	`<let name="str1" type="string">  test trim left</let>`
	`<function name="string.trimleft" > str1 </function>`
	Result:
	str1 = "test trim left"

Function name	Meaning

String.trimright	The function trims the closing characters from a string.
	Parameter:
	str1 - string variable
	Syntax:
	`<function name="string.trimright" > str1 </function>`
	Example:
	`<let name="str1" type="string"> test trim right </let>`
	`<function name="string.trimright" > str1 </function>`
	Result:
	str1 = "test trim right"

sin	The function calculates the sine of the value transferred in degrees.
	Parameter:
	double - angle
	Syntax:
	`<function name="sin" return="<double val>"> double </function>`
	Example:
	`<let name= "sin_val" type="double"></let>`
	`<function name="sin" return="sin_val"> 20.0 </function>`

cos	The function calculates the cosine of the value transferred in degrees.
	Parameter:
	double - angle
	Syntax:
	`<function name="cos" return="<double val>"> double </function>`
	Example:
	`<let name= "cos_val" type="double"></let>`
	`<function name="cos" return="cos_val"> 20.0 </function>`

tan	The function calculates the tangent of the value transferred in degrees.
	Parameter:
	double - angle
	Syntax:
	`<function name="tan" return="<double val>"> double </function>`
	Example:
	`<let name= "tan_val" type="double"></let>`
	`<function name="tan" return="tan_val"> 20.0 </function>`

Function name	Meaning

arcsin	The function calculates the arcsine of the value transferred in degrees.
	Parameter:
	double - x in the range from -PI/2 to +PI/2
	Syntax:
	`<function name="arcsin" return="<double val>"> double </function>`
	Example:
	`<let name= "arcsin_val" type="double"></let> <function name="arcsin" return=" arcsin_val"> 20.0 </function>`

arccos	The function calculates the arccosine of the value transferred in degrees.
	Parameter:
	double - x in the range from -PI/2 to +PI/2
	Syntax:
	`<function name="arcos" return="<double val>"> double </function>`
	Example:
	`<let name= "arccos_val" type="double"></let> <function name="arccos" return=" arccos_val"> 20.0 </function>`

arctan	The function calculates the arctan of the value transferred in degrees.
	Parameter:
	double - arctan of y/x
	Syntax:
	`<function name="arctan" return="<double val>"> double </function>`
	Example:
	`<let name= "arctan_val" type="double"></let> <function name="arctan" return="arctan_val"> 20.0 </function>`

dll.load	The function loads an additional user DLL to the memory.
	Parameter:
	dll_name - DLL name
	class_name - name of the function class
	Syntax:
	`<function name="dll.load"> dll_name, class_name </function>`
	Example:
	`<function name="dll.load"> _T"customer.dll", _T"customer" </function>`

Function name	Meaning

dll.function	The function calls a function from a user DLL. All parameters listed after the parameter ID are transferred to the function called.
	Parameter:
	class_name - name of the function class
	id - of the function
	parameter - maximum seven function parameters (string variables)
	Syntax:
	`<function name="dll.function"> class_name, id, parameter1, parameter2</function>`
	Example
	`<function name="dll.function"> _T"customer", 290, _T"par1", _T"par2"</function>`

File processing	

doc.readfromfile	The function loads the contents of the file specified to a string variable.
	Attribute:
	Return - name of the local variable
	Parameter:
	Progname - file name
	Syntax:
	`<function name="doc.readfromfile" return="<string var>"> progname </function>`
	Example:
	`<let name = "my_var" type="string" ></let>`
	
	`<function name=" doc.readfromfile " return="my_var"> _T"\spf\test.mpf" </function>`

doc.writetofile	The function writes the contents of a string variable to the file specified.
	Parameter:
	progname - file name
	str1 - string
	Syntax:
	`<function name="doc.writetofile" > progname, str1 </function>`
	Example:
	`<let name = "my_var" type="string" > file content </let>`
	
	`<function name="doc.writetofile">_T"\spf\test.mpf", my_var </function>`

Function name	Meaning

doc.remove	The function removes the file specified from the directory.
	Parameter:
	progname - file name
	Syntax:
	`<function name="doc.remove" > progname </function>`
	Example:
	`<function name="doc.remove">_T"\mpf\test.mpf" </function>`

doc.exist	If the file exists, the function returns the value 1.
	Parameter:
	progname - file name
	Syntax:
	`<function name="doc.exist" return="<int_var>" > progname </function>`
	Example:
	`<let name ="exist">0</let>`
	
	`<function name="doc.exist" return="exist">_T"\mpf\test.mpf" </function>`

ncfunc.select	The function selects the program specified for execution. The program must be stored in the NC file system.
	Parameter:
	progname - file name
	Syntax:
	`<function name="ncfunc.select"> progname </function>`
	Example:
	`<function name="ncfunc.select"> _T"\mpf\test.mpf" </function>`

Licensing in SINUMERIK 808D 17

17.1 Licensing in SINUMERIK 808D

SINUMERIK 808D licensing

The PPU software on the CNC PPU has already been licensed in the factory before delivery.

Depending on specific requirements, factory licensing is available for the following machining types:

- SINUMERIK 808D Turning
- SINUMERIK 808D Milling

You can also purchase the optional "**Manual Machine Plus**" and "**Additional Axis**" functions for the SINUMERIK 808D Turning. You must activate these two functions on the control system via the HMI user interface.

Note

You can obtain subsequent licenses from the Web License Manager (http://www.siemens.com/automation/license).

17.2 Web License Manager

17.2.1 Web License Manager

With the Web License Manager (http://www.siemens.com/automation/license) , you can assign licenses to hardware in a standard Web browser. To conclude the assignment, you must manually enter the License Key at the control system through the HMI user interface.

17.2.2 Assigning licenses

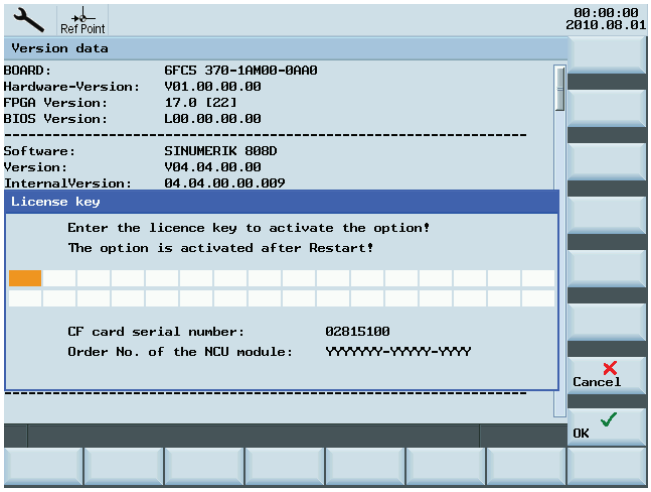
Requirements

The following prerequisites must be met in order to assign a license to a piece of hardware via direct access and HMI user interface:

- The control system is powered up.
- The login data for direct access (e.g. per CoL) is available:
  - License number
  - Dispatch note number
- Type of the control system
- "CF card serial number" from the CompactFlash Card system

Operating sequences

1. In the "SYSTEM" operating area, press "Serv. displ." > "Version" > "License key" to find the "CF card serial number".

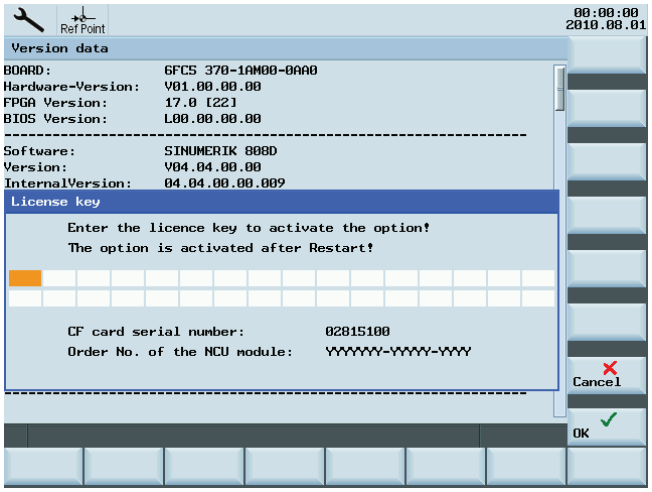


Note

Ensure that the "CF card serial number" displayed is also really the one you want to make the assignment for. The assignment of a license to a piece of hardware cannot be reversed via the Web License Manager.

2. Go to the Web License Manager (http://www.siemens.com/automation/license):
3. Login via "Direct access":
   - License number
   - Dispatch note number
4. In the Web License Manager, follow the instructions and operate step by step.
5. At the end of the process, the Web License Manager shows the license key.

   The following options are now available:
   - Note down the key.
   - Save it in a PDF file.
6. After completing the assignment process, enter the License Key displayed on the Web License Manager into the licensing dialog of the HMI user interface (In the "SYSTEM" operating area, press "Serv. displ." > "Version" > "License key".)



7. Press the "OK" softkey to confirm the entry for the new license key.
8. Activate the optional functions **Manual Machine Plus** and **Additional Axis**.

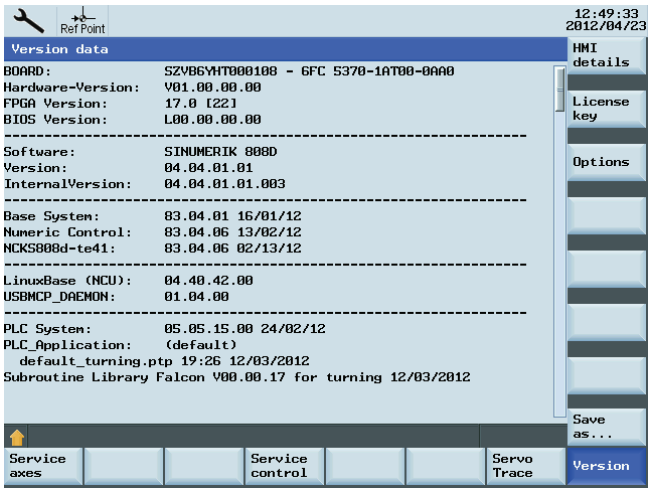
Note

For detailed information about activating the optional functions, refer to section "Activating the optional functions (Page 278)".

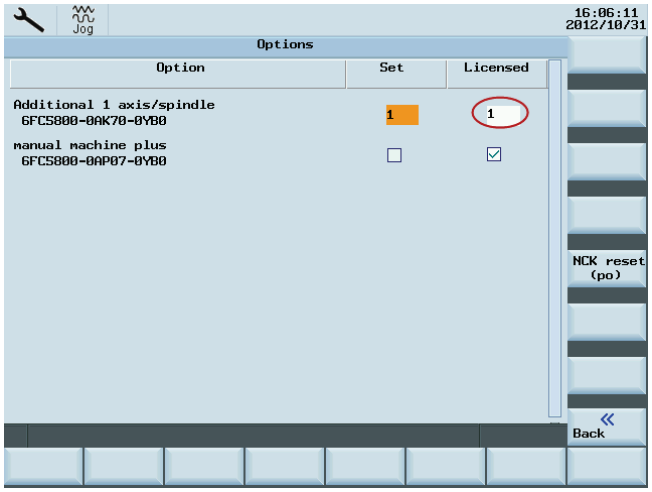
17.3 Activating the optional functions

To activate the optional **Manual Machine Plus** and **Additional Axis** functions for the turning variant of the SINUMERIK 808D, proceed as follows:

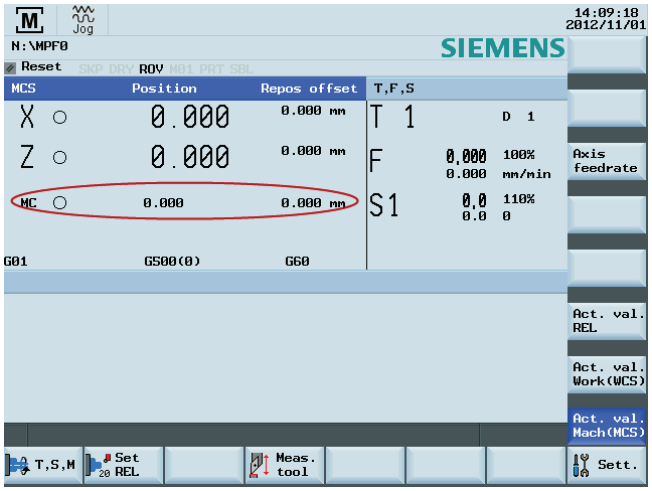
1. In the "SYSTEM" operating area, press "Serv. displ." > "Version". It displays as follows (example):



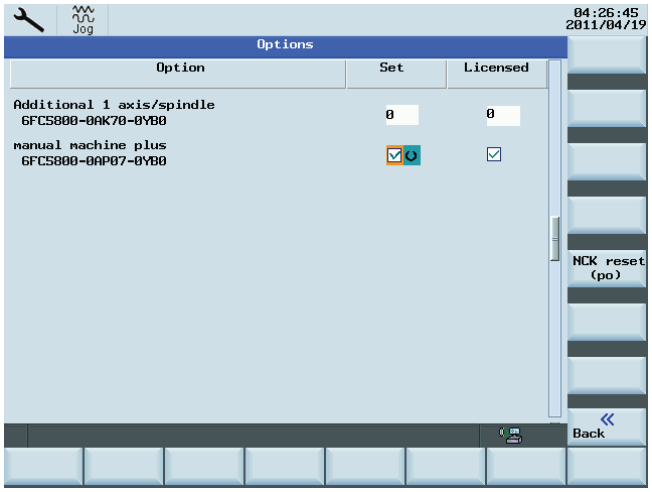
2. Press "Options". If you have activated the **Additional Axis** function by entering the licence key, the value of the **Licensed** column is **1**. Then you can set the **Set** column to **0** or **1**, which respectively indicates that 0 or 1 additional axis is set.



After you assign one additional axis and set the relevant MD correctly (see the note below), the additional axis name can be seen in some operating areas. For example, the "MACHINE" operating area will be shown as follows:



Or, press <SELECT> to select the software option **Manual Machine Plus**.



3. Press "NCK reset (po)". A warm restart is triggered on the control system. After system restart, the corresponding software function is active in the control system.

Note

If either of the two options is not licensed, the alarm 8081 "1 option(s) that has (have) not been licensed using a license key was (were) set" appears on the screen. Enter the valid license; otherwise, you cannot operate the machine normally.

To use the additional axis function after activating it, set the following MDs first:

- MD10000[1]: Change "**Y**" to another symbol except capital letter **C**, for example, "**MC**".
- MD20070[3]: Change "**0**" to "**2**".
- MD20080[3]: Set the value to "**C**".
- When necessary, set the following MDs , wherein MD30300 determines whether the additional axis is used as a rotary or linear axis.
  - MD30300
  - MD30310
  - MD30320

17.4 Internet links

Overview of Internet links used:

No.	Topic	Address
1	Web License Manager	http://www.siemens.com/automation/license
2	Siemens A&D Mall: Customer login	http://mall.automation.siemens.com
3	Download server	http://software-download.automation.siemens.com

17.5 Important licensing terms

The terms below are important and helpful for you to understand the license management of SINUMERIK software products.

Term	Description
Software product	"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.
Hardware	In the context of the 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. License information is also saved to the retentive memory on this component. • SINUMERIK 808D: CompactFlash Card system
License	A license gives the user a legal right to use the software product. Evidence of this right is provided by the following: • CoL (Certificate of License) • License key
CoL (Certificate of License)	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 **Note:** The hardware serial number is only found on a system software CoL or is only available if a bundled license was ordered, in other words, the system software included options.
License number	The license number is the feature of a license that is used for its unique identification.
CompactFlash Card system	The CompactFlash Card system represents, as the carrier of all the retentive data of a SINUMERIK solution line control system, the identity of this control system. The CompactFlash Card system includes the following data that is of relevance to license management: • Hardware serial number • License information including the License Key

Term	Description
Hardware serial number	The hardware serial number is a permanent part of the CompactFlash Card system. It is used to identify a control system uniquely. The hardware serial number can be determined by: • CoL (see: Certificate of License > "Note") • HMI user interface ("SYSTEM" operating area > "Serv. displ." > "Version" > "License key") • Printing on the CompactFlash Card system
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	One 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.
Product	A product is marked by the data below within the license management of SINUMERIK software products: • Product designation • Order number: • License number

Index

I

J

L

M

O

P

R

S

T

V

X